robotron
datenbank-software

HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

DIPLOMARBEIT

Herr
Steffen Klauck

System zur Verwaltung von Testfällen für die Produktentwicklung robotron*ecount

2010

Fakultät **Informationstechnik & Elektrotechnik**

DIPLOMARBEIT

System zur Verwaltung von Testfällen für die Produktentwicklung robotron*ecount

Autor:
Steffen Klauck

Studiengang:
Multimediatechnik

Seminargruppe:
MK06w1

Erstprüfer:
Prof. Dr. rer. biol. hum. Stübner

Zweitprüfer:
Dipl.-Inf. Björn Heinemann

Dresden, Oktober 2010

Bibliografische Angaben

Klauck, Steffen: System zur Verwaltung von Testfällen für die Produktentwicklung robotron*ecount, 81 Seiten, 43 Abbildungen, Hochschule Mittweida, Fakultät Informationstechnik & Elektrotechnik

Diplomarbeit, 2010

Abstract

Within the framework of this thesis, for the development of the energy data management system robotron*ecount it is examined how a system for the management of test cases can be integrated into the existing infrastructure. Basic principles associated with test case management are explained. First, a requirement analysis is conducted. Based on this, an evaluation of the systems available on the market is carried out. On the basis of these evaluations, a strategy for how an own creation of a program can be integrated into the existing infrastructure is created. After this, the strategy is implemented prototypically. The implementation is evaluated in terms of the requirements.

Inhaltsverzeichnis

I. Abbildungsverzeichnis

II. Abkürzungsverzeichnis

BLOB ..... Binary Large Object, Seite 17

CASE ..... Computer Aided Software Engineering, Seite 11

CAST ..... Computer Aided Software Testing, Seite 11

FV ..... Fehlerverwaltung, Seite 40

ID ..... Identifikation, Seite 45

IEEE ..... Institute of Electrical and Electronics Engineers, Seite 6

IT ..... Informationstechnologie, Seite 8

JDBC ..... Java Database Connectivity, Seite 17

KBSt ..... Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung, Seite 8

KISS ..... Keep it simple and smart, Seite 17

KM ..... Konfigurationsmanagement, Seite 14

Lampp ..... Linux Apache MySQL PHP Perl, Seite 23

Oracle ADF ..... Oracle Application Development Framework, Seite 39

PHP ..... Hypertext Preprocessor, Seite 23

PL/SQL ..... Procedural Language/SQL, Seite 17

Rth ..... Requirements and Testing Hub, Seite 31

SPU ..... Software-Produktionsumgebung, Seite 12

SQL ..... Structured Query Language, Seite 17

SUT ..... System Under Test, Seite 7

TFVS ..... Testfallverwaltungssystem, Seite 40

WYSIWYG ..... What You See Is What You Get, Seite 17

1 Einleitung

Die Robotron Datenbank-Software GmbH wurde im Jahr 1990 gegründet. Das Tätigkeitsfeld des Unternehmens ist die Entwicklung von Datenbank-gestützten Informationssystemen, die zur Verwaltung und Auswertung großer Datenmengen benutzt werden können. Im Bereich der Energiedatenmanagement-Systeme hat sich die Robotron Datenbank-Software GmbH mit dem Produkt robotron*ecount als ein führendes Unternehmen etabliert.

Ziel der Diplomarbeit ist es, für die Entwicklung des Energiedatenmanagement-Systems robotron*ecount zu untersuchen, wie ein System zur Verwaltung von Testfällen in die vorhandene Infrastruktur integriert werden kann. Dazu soll eine Anforderungsanalyse durchgeführt werden. Darauf aufbauend soll eine Evaluation der auf dem Markt vorhandenen Systeme erfolgen. Auf der Grundlage dieser Auswertungen ist ein Konzept zu erstellen, wie ein System in die vorhandene Infrastruktur integriert werden kann. Im Anschluss soll das Konzept prototypisch umgesetzt werden.

Durch Software-Prüfung sind 70 bis 90 % weniger Fehler in der ausgelieferten Software und 10 bis 30 % niedrigere Gesamtkosten bei 5 bis 20 % kürzeren Entwicklungszeiten zu erwarten. [Fru07]

Testen bewirkt Einsparungen durch die Aufdeckung von Fehlerwirkungen und anschließende Beseitigung der Fehlerzustände in der Software. Fehlerzustände, die hier nicht entdeckt werden, verursachen im Betrieb meist hohe Kosten. Das Thema Softwaretest bleibt immer aktuell, da ein Softwareprodukt im Laufe seines Lebenszyklusses fortwährend Tests unterzogen wird. Nach der Fertigstellung des Produkts müssen Tests für Änderungen durch Erweiterungen, Aktualisierungen und Fehlerbehebungen durchgeführt werden.

Eine Ableitung des Paretoprinzips, auch 80-zu-20-Regel genannt, besagt: „20 % der Fehler verursachen 80 % der Kosten.“ [Lig09, S. 28] Das Vermeiden dieser 20 % der Fehler ist das Ziel von Qualitätsverbesserungen.

Die Qualität der Umsetzung wachsender Kundenanforderungen an komplexe Systeme soll gesichert werden. Durch eine Verbesserung des Qualitätsmanagements sind erhebliche Kosteneinsparungen möglich. Das Testen von Software ist ein elementarer Teil der Qualitätssicherung. Der Entwurf von Testfällen ist ein wichtiger Teil des Testprozesses. Testen ist mit einem hohen Arbeitsaufwand verbunden und muss hauptsächlich per Hand durchgeführt werden. Die qualitative und quantitative Auswahl von Testfällen entscheidet darüber, ob vorhandene Fehler im Testgegenstand gefunden werden und mit welchen Kosten das Finden dieser Fehler verbunden ist.

Für die Produktentwicklung robotron*ecount wird ein System gesucht, das es ermöglicht, aus einer großen Anzahl definierter Testfälle die zu einer Anforderung passenden herauszufinden, auszuführen und zu dokumentieren. Dadurch soll der wirtschaftliche und produktive Nutzen des Testens gesteigert und die Arbeit der Qualitätssicherung unterstützt werden. Die Aufgabe wurde als Diplomarbeit ausgeschrieben. In der Arbeit wird die methodische und technische Vorgehensweise der Software-Prüfung identifiziert, um Anforderungen an ein Testmanagementsystem zu ermitteln. Daraufhin werden auf dem Markt vorhandene Systeme evaluiert. Auf Grundlage dieser Evaluation wird ein Konzept für den Einsatz eines Systems entwickelt. Das Konzept wird prototypisch umgesetzt und bezüglich der Anforderungen evaluiert. Die Erstellung eines Einführungskonzepts für das Werkzeug ist nicht Bestandteil der Diplomarbeit.

Im folgenden Kapitel wird das Unternehmen Robotron Datenbank-Software GmbH vorgestellt, in dem die Diplomarbeit erarbeitet wurde.

2 Robotron Datenbank-Software GmbH

In diesem Kapitel wird die Robotron Datenbank-Software GmbH vorgestellt. Im Anschluss werden ihre Produkte für den Energiemarkt beschrieben.

2.1 Allgemein

Das Software-Unternehmen Robotron entwickelt Datenbank-gestützte Informationssysteme. Als Partner der Oracle Corporation werden vorrangig Oracle-basierte kundenspezifische Anwendungslösungen geboten.

Robotron ist unter anderem in den Bereichen Energiedatenmanagement, Museumsdatenmanagement, Vorgangsbearbeitung bei der Polizei und Fördermittelverwaltung tätig.

Der Hauptsitz des Unternehmens ist in Dresden. Dort arbeiten mehr als 200 Mitarbeiter. Robotron besitzt Tochtergesellschaften in der Schweiz, Österreich, der Tschechischen Republik und Russland.

2.2 robotron*ecount

Da die produktionsbedingte Standortgebundenheit auf dem Energiemarkt zur Monopolbildung führen kann, wurde das Energiewirtschaftsgesetz erlassen. So wurden geschlossenen Versorgungsgebiete beseitigt und eine strikte Trennung zwischen Teilen der Lieferkette, zum Beispiel der Energieerzeugung und dem Netzbetrieb, eingeführt. Die Trennung muss auch auf der Ebene der Softwaresysteme eingehalten werden. Durch die Liberalisierung des Strom- und Gasmarktes wurde es notwendig zu ermitteln, welcher Lieferant wie viel Energie an seine Kunden geliefert hat. Mit robotron*ecount können die großen Datenmengen, die Zähler täglich ermitteln, verarbeitet und ausgewertet werden. Zeitreihen- und Stammdaten können beliebig berechnet und visualisiert werden. Das Produkt bietet beispielsweise Möglichkeiten zur Sicherung von Datenkonsistenz- und Qualitätsparametern, Protokollmechanismen, Datenimport und -export, Präsentation von Ergebnismengen, Nachvollziehbarkeit von wichtigen Datenänderungen und kontinuierliche Überwachung des Systemzustandes. Routinearbeiten können automatisiert werden, sodass nur noch Problemfälle betrachtet werden müssen. Robotron*ecount stellt Schnittstellen bereit, über deren Benutzung der tägliche Arbeitsaufwand für den Datenaustausch mit anderen Marktteilnehmern (Kunden, Händlern und Netzbetreibern) verringert wird. Sicherheitsanforderungen werden erfüllt. Dazu zählen die Transaktionssicherheit, die Sicherung der Datenbestände und der Schutz vor Missbrauch der über Schnittstellen zugänglichen Architekturkomponenten.

Robotron*ecount beruht auf dem Datenbankmanagementsystem von Oracle. Oracle Forms bildet die Basis für die Benutzeroberfläche.

Robotron*ecount lässt sich im Funktionsumfang und Umfang der zu verwaltenden Daten an die Erfordernisse des Kunden anpassen. Den Kunden wird ein Schulungsprogramm angeboten. Die Kurse reichen von den Grundlagen für Anwender bis zu individuellen Workshops zu speziellen Themen.

2.3 robotron*esales

Durch robotron*esales wird robotron*ecount um Funktionen erweitert, die Bereiche der Beschaffung und des Vertriebs abdecken. Robotron*esales kann als eigenständiges System bezogen werden. Als Bestandteil von robotron*ecount können die Funktionen alternativ zur Verfügung gestellt werden. Die unterstützten Funktionen sind in der Abbildung 2.1 dargestellt.

Auch robotron*esales beruht auf dem Datenbankmanagementsystem von Oracle. Oracle Forms wird als Präsentationsschicht verwendet.

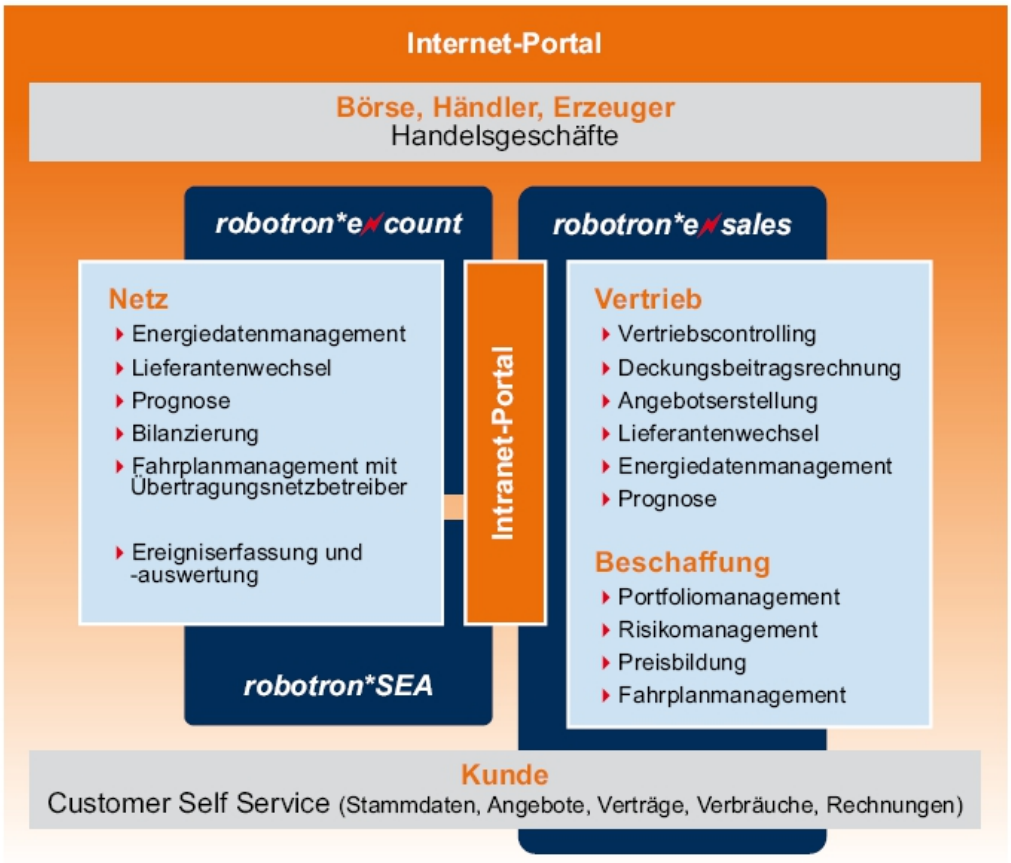


Abbildung 2.1: Funktionsumfang robotron*ecount und robotron*esales [RDS10]

2.4 robotron*ecollect

Robotron*ecollect ist ein Modul, das in robotron*ecount integriert werden kann. Mit dem Modul wird Zählerfernauslesung über die Anbindung von Kommunikationsservern realisiert. Dafür notwendige Funktionen werden bereitgestellt. Eine zeitgesteuerte Bearbeitung von Auslesevorgängen wird ermöglicht und automatisch von den Zählern gesendete Daten werden erfasst. Die ermittelten Daten werden in die Zeitreihenverwaltung importiert und stehen somit unmittelbar den Prozessen des Energiedatenmanagement-Systems robotron*ecount zur Verfügung.

Die Entwicklung des Produkts robotron*ecount bildet den Rahmen der praktischen Tätigkeit dieser Diplomarbeit.

3 Grundlagen

Im Folgenden werden verwendete Begriffe definiert und Grundlagen, die mit einer Testfallverwaltung in Zusammenhang stehen, erläutert.

3.1 Testen

Durch die als Testen bezeichnete Tätigkeit sollen Fehler in einem Programm aufgespürt werden. Es soll eine Erwartung belegt oder widerlegt werden. Der Standard IEEE 829-1998 [IEE98] definiert Testen als:

> „The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item.“

Eine Software wird in Bezug auf die an sie gestellten Anforderungen geprüft. Fehler und Abweichungen werden festgestellt.

Das Verhältnis zwischen dem, was getestet werden kann und dem, was getestet wurde, wird Deckungsgrad genannt. Der Deckungsgrad wird häufig auf den Programmcode (wie viele Programmteile werden mit den vorhandenen Testfällen abgedeckt) oder auf funktionale Spezifikationen bezogen.

Man unterscheidet zwischen dynamischen und statischen Tests. Für statische Tests werden Programmcode und Dokumente analysiert. Die Ausführung eines Programms zum Finden von Fehlern wird als dynamischer Test bezeichnet. Die Ausführung findet in der Praxis systematisch und stichprobenartig statt. In dieser Diplomarbeit werden dynamische Tests betrachtet.

Tester sind für die Testdurchführung und Testprotokollierung verantwortlich. Entdeckte Fehlerwirkungen werden dokumentiert und den Entwicklern gemeldet. Diese ermitteln und beseitigen die Ursachen. Ein erneuter Test muss nachweisen, dass die Fehlerwirkungen nicht mehr auftreten und durch die Änderungen keine neuen hinzugekommen sind.

Tests finden meist in simulierten Umgebungen statt. Daraus folgt, dass die Ergebnisse unter anderem von der Qualität der Simulation abhängig sind.

Ein Testprotokoll, auch Testfallbericht genannt, enthält das Ergebnis eines Testlaufs. Aus dem Protokoll muss hervorgehen, was wann, von wem, wie intensiv, in welcher Umgebung, mit welchen Testmitteln (zur Definition von Testmitteln siehe Abschnitt 3.6) und

mit welchem Ergebnis getestet wurde. [Spi10, S. 28] Das Testprotokoll dokumentiert das Ist-Verhalten der getesteten Software.

Nach [IEE98] ist der Testbericht ein Dokument, das die Testaktionen und -ergebnisse zusammenfasst und die Qualität des Objekts anhand der Testendekriterien bewertet. Der Testbericht wird zum Ende eines Testzyklus erstellt und verteilt. [Spi08, S. 147] Die Vorlage eines Testberichts kann man im Anhang A finden. Sie steht unter [IAG10] zum Herunterladen bereit.

Um die Auswertung von Testfällen ökonomisch vornehmen zu können, wird eine entsprechende Werkzeugunterstützung benötigt. Der Einsatz von Testwerkzeugen erzielt nur den gewünschten Einsparungseffekt, wenn ein Testprozess als solcher beherrscht wird und bereits definiert abläuft.

Die Grundlagen des modellbasierten Testens werden in dieser Arbeit nicht behandelt. Nähere Informationen findet man unter [MOT10] .

3.2 Testprozess

Der Prozess des Testens von Software ist ein Teilprozess der Softwareentwicklung. Er ist notwendig, um eine Aussage über die Qualität eines erstellten Produktes treffen zu können. Hohe Qualität bedeutet, dass den Erwartungen von Anwendern entsprochen wird. Damit ist hohe Qualität mit der Aussage „den Erwartungen entsprechend“ gleichgesetzt. Einige allgemein gültige Standards im Bereich des Testens von Software sollen im Folgenden aufgezeigt werden. Danach wird der Testprozess in der Produktentwicklung robotron*ecount beschrieben.

Ein gut strukturierter und verlässlicher Testprozess innerhalb der Softwareentwicklung ist notwendig, da die Bedeutung von Software in unserer Gesellschaft immer größer wird und ein hohes Budget für die Fehlerbehebung und das Testen ausgegeben wird. Die Qualität von Software ist zu einem wichtigen Überlebensfaktor vieler Firmen geworden.

Der Testprozess besteht aus den Phasen Planung, Spezifikation, Durchführung, Protokollierung und Bewertung. Der Durchlauf durch die Schritte stellt einen Testzyklus dar, an dessen Ende Fehlerkorrektur- oder Änderungsaufträge für den Entwickler vorliegen.

Neben der Einhaltung des Testprozesses sind Reviews qualitätssichernde Maßnahmen. Ein Review ist eine manuelle Prüfmethode, die in einer Teamsitzung Stärken und Schwächen eines Prüfobjekts identifiziert. [MKN10] Das Prüfobjekt ist das zu testende System, auch als System Under Test (SUT) bezeichnet. Mehrere Augenpaare sehen mehr als eins. Reviews nehmen diesen Grundsatz auf. Die Ergebnisse werden im Testbericht, wie in Kapitel 3.1 beschrieben, dokumentiert. Dieser schildert das Ergebnis der Ausführung des

Prüfplans.

Testaktionen können in das Wasserfallmodell, das V-Modell und das W-Modell eingeordnet werden. Im Wasserfallmodell finden Aktionen in folgendem Ablauf statt: Anforderungsanalyse, Design, Implementierung, Test, Wartung. Nach dem V-Modell finden Tests in mehreren, aufeinander folgenden Phasen statt.

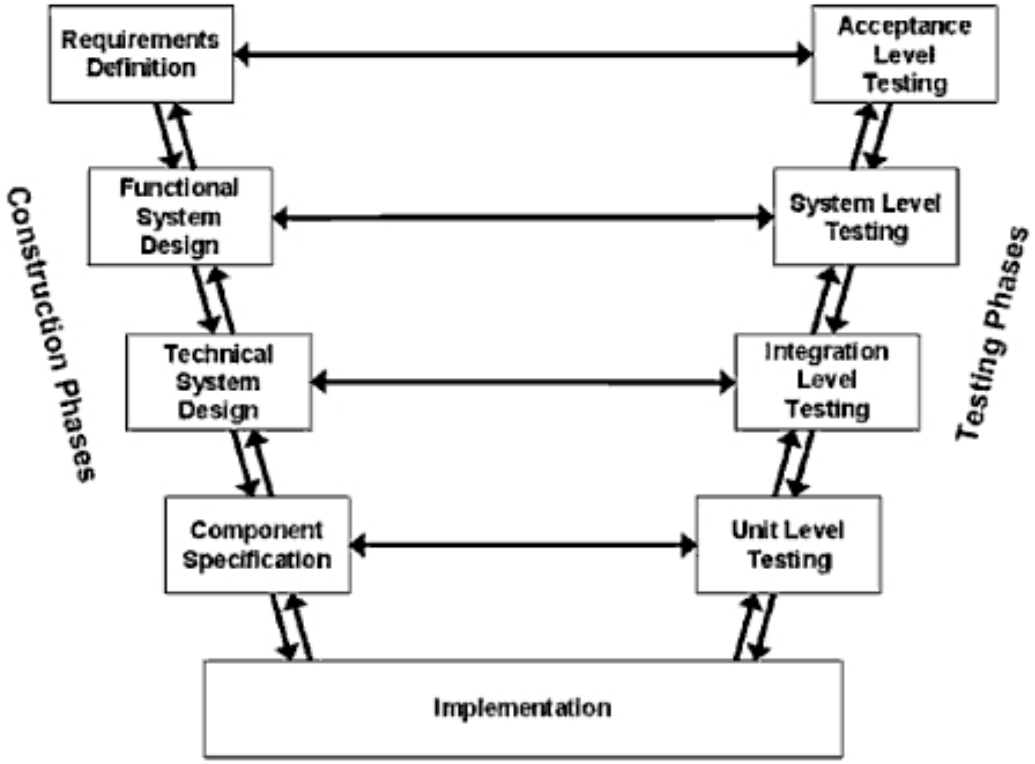


Abbildung 3.1: V-Modell [MOT10]

Bei Robotron ist das V-Modell, Vorgehensmodell der KBSt (Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung) und Entwicklungsstandard für IT-Systeme des Bundes, festgelegt.

Weiterhin kann eine Einordnung von Testaktionen wie in Abbildung 3.2 erfolgen. Es wird zwischen Einzeltests, Integrationstests und Abnahmetests unterschieden.

Umgebung	Zweck	Konfiguration	Datenhaltung	Verantwortung
Entwicklungs-umgebung	○ Entwicklung ○ Einzeltest	auf Bedürfnisse der Entwicklung ausgelegt	synthetische Testdaten in Verantwortung der Entwickler	Entwicklung
Integrations-umgebung	○ Integrations-arbeiten ○ Integrations-tests	Software produktionsnah, Hardware einfacher	synthetische Daten	Integration
Vorproduktion	○ Systemtests ○ Abnahmetests	im Wesentlichen wie Produktion	Datenabzug aus Produktion	Betrieb
Produktion	○ *evtl. Leistungs-messung*	*produktive Konfiguration*	*produktive Daten*	*Betrieb*

Abbildung 3.2: Testumgebungen mit ihren typischen Eigenschaften [Fru07, S. 124]

Im für die Produktentwicklung robotron*ecount festgelegten Prozess stellen Melder und Projektleiter Testfälle im Rahmen der Konzepterstellung bereit. Diese existieren vor der Entwicklung. Der Unterschied zur Theorie der testgetriebenen Entwicklung besteht darin, dass während der Produktentwicklung robotron*ecount häufig nicht der Entwickler die Testfälle erstellt. Dadurch wird die Nachlässigkeit und mangelnde Disziplin der Programmierer bei der Testerstellung umgangen. Weiterhin werden so Fehlinterpretationen der Entwickler von Anforderungen und Konzepten durch die unabhängigen Interpretationen der Tester erkannt. Nach der Entwicklung ist eine Anpassung der Prüfspezifikationen möglich. Es finden Alphatests durch Robotron parallel zur Entwicklung und vor den Releases statt. Im Anhang B sind die Testaktionen während der Produktentwicklung robotron*ecount im Fall einer Fehlerbearbeitung dargestellt [Quelle: Robotron].

3.3 Testfall

Während des Testens werden Testfälle durchgeführt. Ein Testfall beschreibt Vorbedingungen, Eingaben oder Handlungen und erwartete Ausgaben.

Das Ergebnis eines Testfalls kann nur als fehlerhaft eingestuft werden, wenn vorab festgelegt wurde, wie das erwartete Ergebnis aussieht. Erwartete Ergebnisse können während der Erstellung von Testfällen aus dem Testorakel hergeleitet werden. Das Orakel dient als Informationsquelle. Anforderungsdefinitionen, formale Spezifikationen und das Benutzerhandbuch des Testobjekts sind Beispiele hierfür. [Spi08, S. 5]

Man unterscheidet zwischen logischen und konkreten Testfällen. In logischen Testfällen werden meist Wertebereiche für die Eingaben und Ausgaben angegeben. [Spi10, S. 24] Logische Testfälle werden durch die Festlegung tatsächlicher Eingabewerte, Ausgabe-

werte und Rahmenbedingungen zu konkreten Testfällen.

Durch den Einsatz von Testfällen und der daraus resultierenden Notwendigkeit, diese häufig an Änderungen und neue Entwicklungen anzupassen, ergibt sich ein Mehraufwand.

Mehr Testfälle bedeutet nicht zwangsläufig bessere Tests. Dieser Gedanke soll bei der Testfallerstellung Beachtung finden.

3.4 Testmanagement

Unter Testmanagement versteht man nach [Spi10, S. 264] :

1. „Planung, Aufwandsschätzung, Steuerung und Auswertung von Testaktivitäten, die typischerweise durch einen Testmanager erfolgen.
2. Verantwortliche Person(engruppe) für den Test.“

Das Testmanagement gibt Richtlinien vor, die beim Testen umgesetzt werden. Ergeben sich beim Testen Abweichungen, greift das Testmanagement steuernd ein.

Durch die Verwaltung von Tests soll eine sogenannte Requirement Test-Matrix aufgestellt werden können. Durch diese kann kontrolliert werden, wie viele Anforderungen durch vorhandene Tests abgedeckt sind.

3.5 Testmanagementsysteme

Testmanagementsysteme können zur Verbesserung der Kommunikation, Organisation und Dokumentation von wiederholbaren Testprozessen dienen. Sie verwalten den Vorgang, verkürzen den Prozess und erlauben es, Ressourcen gezielter einzusetzen. Durch den Einsatz von Testmanagementsystemen können Testaufgaben effizienter und präziser bearbeitet werden. Die Testabdeckung kann mit gleichem Ressourcenaufwand erhöht werden. Testfälle können spezifiziert und ausgeführt, und deren Ergebnisse ausgewertet und dokumentiert werden. Zunächst folgt eine Einordnung von Testmanagementsystemen.

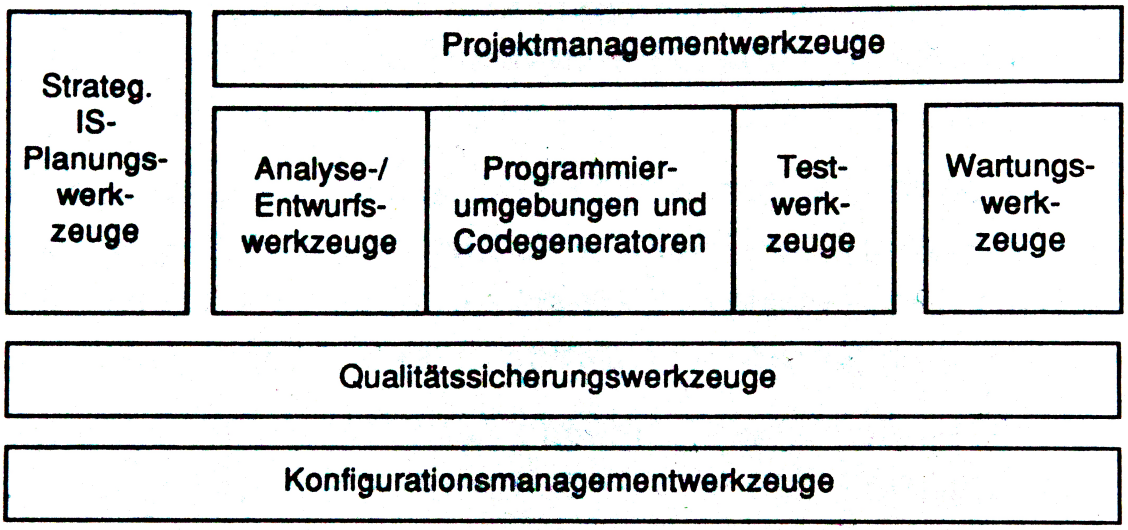


Abbildung 3.3: CASE-Werkzeugklassifikation [Wal90, S. 110]

Ausgehend von der Abbildung 3.3 werden in dieser Diplomarbeit Werkzeuge der Qualitätssicherung betrachtet. Man unterscheidet hier zwischen Werkzeugen zur Fehlerverhütung, -erkennung und -auswertung. Testmanagementsysteme zählen zu Werkzeugen zur Fehlererkennung und Fehlerauswertung. Werkzeuge für die Verwaltung von Testfällen und Tests erlauben es, Testfallspezifikationen geordnet abzulegen, zu priorisieren und ihren Status zu überwachen. Die Beziehung zu den Anforderungen kann verwaltet werden, sodass Lücken sichtbar werden. Die Ergebnisse der Testausführung können erfasst werden. [Fru07, S. 121] Beispiele sind:

- IBM Rational Quality Manager
- Requirements and Testing Hub

Testmanagementsysteme lassen sich auch unter dem Begriff CAST einordnen. Der Begriff CAST ist eine Abwandlung des Begriffs CASE.

CASE (Computer Aided Software Engineering) ist der Entwicklungs- und Pflegeprozess von Software, unterstützt durch Werkzeuge. [Wal90, S. 110] In den letzten Jahren hat sich die Bezeichnung „Entwicklungsumgebung“ für diese Werkzeuge durchgesetzt.

Werkzeuge, die das Automatisieren von Testaktionen ermöglichen, werden in Anlehnung an den Begriff CASE unter dem Begriff CAST-Tools (Computer Aided Software Testing) zusammengefasst. [Spi10, S. 209] In dieser Diplomarbeit werden Werkzeuge für das Management und die Steuerung von Tests betrachtet. Diese bieten Mechanismen zur Erfassung, Priorisierung, Katalogisierung und Verwaltung von automatischen und manuellen Testfällen. Die Testfallausführung wird überwacht und Ressourcen- und Zeitplanung unterstützt. Viele Testmanagementwerkzeuge bieten zusätzlich eine Fehler-

verwaltung, die Erstellung von Testberichten und -dokumenten und andere Funktionen. Weitere Werkzeugtypen, die unter den Begriff CAST eingeordnet werden, sind:

- Werkzeuge zur Testdaten- und Testspezifikation
- Werkzeuge für den statischen Test (Code-Analyse)
- Werkzeuge für den dynamischen Test (die Software wird ausgeführt)

Neben den Werkzeugen für die Verwaltung von Testfällen und Tests, Testtreibern, Dateivergleichern und Testdatengeneratoren schaffen Compiler und Werkzeuge für die Versions- und Konfigurationssteuerung wichtige Voraussetzungen für Tests.

Den kreativen Kern der Testarbeit kann kein Werkzeug wirkungsvoll unterstützen. Routineaufgaben dagegen können erheblich erleichtert werden. [Fru07, S. 120] Unersetzlich für den Testmanager ist ein Werkzeug zur Verwaltung der Problem- beziehungsweise Fehlermeldungen. Oft sind Fehlermanagement- und Testmanagementwerkzeuge miteinander gekoppelt.

Ein Testmanagementsystem soll in die Software-Produktionsumgebung (SPU) integriert sein. Dabei sind folgende Fragen zu klären:

- Wo fehlt es an Methodenunterstützung im Prozess?
- Wie groß ist die Kluft zwischen benötigter Methodik und der tatsächlich vom Werkzeug unterstützten Methodik? [Wal90, S. 116]

Siehe dazu Abbildung 3.4.

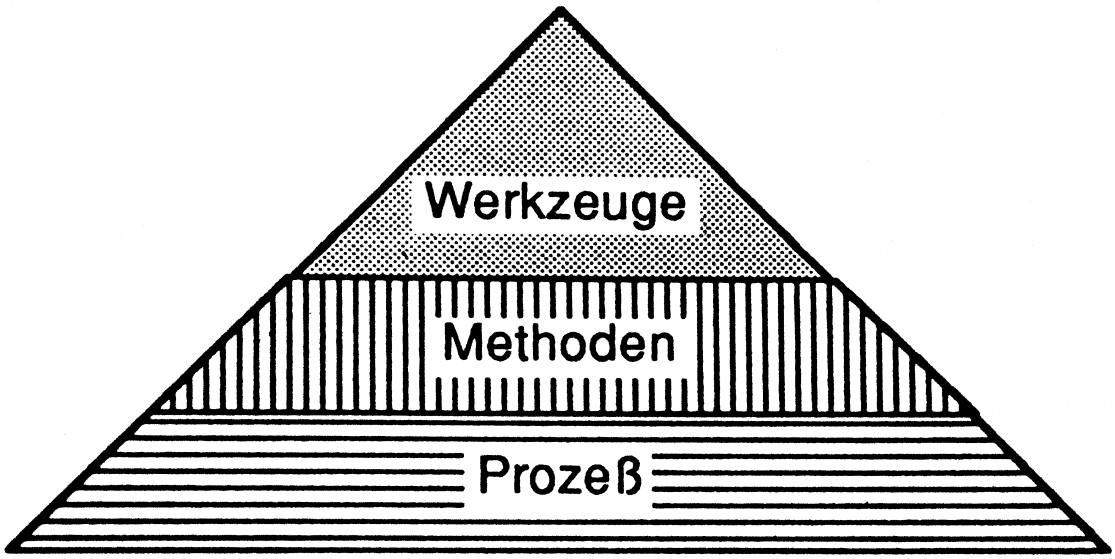


Abbildung 3.4: SPU-Modell nach Charette [Wal90, S. 116]

M. Fewster und D. Graham schreiben in ihrem Buch „Software Automation“:

> „It is far better to improve the effectiveness of testing first than to improve the efficiency of poor testing. Automating chaos just gives faster chaos“

Ein Werkzeug kann nur effektiv eingesetzt werden, wenn wohldefinierte Prozesse und Vorgehensweisen vorhanden sind.

Für die Einführung von Testwerkzeugen muss ein Einsatzkonzept zur Verfügung gestellt werden. Mindestens eine Person ist notwendig, die vom Nutzen des Werkzeugs begeistert ist und die Benutzung beherrscht, sodass sie ihre Kollegen beraten kann. Es muss sichergestellt werden, dass Alle das Werkzeug ähnlich benutzen, mit Kollegen kooperieren, nicht an ihre oder die Grenzen des Werkzeugs stoßen und zu ihrer vertrauten Arbeitsweise zurückkehren. Die Werkzeugeinführung ist eine Investition. Zunächst fallen Kosten für die Auswahl, Anschaffung, Wartung und Mitarbeiterschulung an. Das Investitionsvolumen kann je nach Komplexität des Werkzeugs schnell in den fünfstelligen Bereich gehen. Der finanzielle Nutzen stellt sich später ein. [Fru07, S. 124] Es muss betrachtet werden, inwieweit die Testqualität durch den Einsatz des neuen Werkzeugs ansteigt, wodurch mehr Fehler gefunden und behoben werden. Entwicklungs-, Support- und Wartungskosten werden mittelfristig gesenkt. Neue Tester sollen sich durch den Einsatz von Testmanagementwerkzeugen schneller zurechtfinden.

Die Einführung des Werkzeugs ist nicht Bestandteil der Arbeit.

Der Einsatz von Testrobotern ist wegen der vielen Änderungen der aufgezeichneten Testscripts umständlich. Eine stabile Benutzerschnittstelle ist notwendig. Die Voraussetzung ist in der Produktentwicklung robotron*ecount nicht gegeben. Deshalb besitzt die Möglichkeit der Integration von Robotern in das Testmanagementsystem nur geringe Bedeutung. Abgesehen davon lohnt sich die Aufzeichnung manueller Tests in Einzelfällen dennoch, da Entwicklern damit ein präzises und ausführliches Testprotokoll zur Verfügung steht.

3.6 Test-Frameworks

In Abgrenzung zu Testmanagementsystemen werden hier der Begriff Testrahmen (engl. Test-Framework) und damit im Zusammenhang stehende Begriffe definiert. Dieser Abschnitt besitzt den Charakter einer Aufzählung.

Ein Testrahmen besteht aus allen Programmen, die zum Ausführen, Auswerten und Aufzeichnen der Testfälle notwendig sind. Dazu zählen unter anderem wichtige Methoden, Testtreiber und Dummies. Testrahmen sind meist selbst zu programmieren. Testrahmengeneratoren analysieren die Programmierschnittstelle des Testobjekts und erzeugen einen Testrahmen. Test-Frameworks unterscheiden sich also von Testmanagementsystemen.

Testmittel (engl. testware) ist ein Oberbegriff für Software, die während des Testprozesses erstellt und benutzt wird, jedoch nicht operativen Charakter haben muss. Zu Testmitteln zählen zum Beispiel Testfälle, Testprotokolle, Testinfrastruktur und eingesetzte Werkzeuge. [Spi10, S. 265] Testmittel sollten für Testzyklen nicht immer wieder neu erstellt werden müssen, sondern zur Übertragung und Aktualisierung bereitstehen.

Zur Testinfrastruktur zählen die Arbeitsplätze der Tester und deren Ausstattung, die Testumgebung und sonstige Werkzeuge wie zum Beispiel E-Mail, WWW und Texteditoren. [Spi10, S. 264]

Die Testumgebung ist die Gesamtheit aller Hardware- und Softwarekomponenten. Sie ist notwendig, um Testfälle durchzuführen. Elemente sind beispielsweise Testrahmen mit Testtreibern, Testdaten, Simulatoren und Analysatoren. Die Testumgebung sollte der Produktionsumgebung möglichst ähnlich sein. [WIT10]

Testobjekte sind die einem Test unterzogenen Komponenten, Teilsysteme, Systeme oder Methoden in einer bestimmten Version. [Spi10, S. 265] Ein Testobjekt ist ein Teil des SUT.

Eine Komponente ist ein Teil eines größeren Systems. Für den Test einer Komponente muss daher ein Testgeschirr bereitgestellt werden. Das Geschirr besteht aus Geräten, Maschinen und Vorrichtungen, die für eine bestimmte Arbeit zusammengestellt sind. [Fru07, S. 68] Dazu zählt zum Beispiel eine Datenbank. Während des Befragens der Tester der Produktentwicklung robotron*ecount wurde mir berichtet, dass in der Testdatenbank die Daten ungeordnet vorliegen. Es wäre nötig, jedoch nicht realisierbar, für jeden Kunden eine eigene Datenbank mit spezifischen Daten zu führen.

3.7 Konfigurationsmanagement

Konfigurationsmanagement (KM) ist ein Verfahren zur Sicherstellung der Identifikation und Rückverfolgbarkeit von Produkten. Eine Konfiguration ist eine Menge von Einzelelementen, aus denen ein Produkt in einer speziellen, zu einem bestimmten Zeitpunkt gültigen Ausprägung aufgebaut ist. [PMG10] Ein Konfigurationselement ist jede maschinenlesbare Informationseinheit, die im Laufe eines Softwareprojektes entsteht. [SKM08] Diese wird über ein Konfigurationsmanagementsystem erfasst und vom Konfigurationsmanagementprozess verwaltet. Testdaten und -resultate müssen dem Konfigurationsmanagement unterworfen sein. Dazu zählt das Identifizieren und Definieren von Konfigurationselementen, die Aufzeichnung und der Bericht des Status von Konfigurationselementen und das Überprüfen auf Vollständigkeit und Richtigkeit der Konfigurationselemente.

Weitere Angaben zu Softwarekonfigurationsmanagement kann man unter [SKM08] nach-

lesen.

Bei Robotron wird der Konfigurationsmanagement-Prozessablauf dokumentiert.

3.8 Evaluation

Während der Erarbeitung der Diplomarbeit wurden Testmanagementsysteme evaluiert. Im Folgenden werden allgemeine Grundlagen zur Evaluation beschrieben.

Eine Evaluation ist die Sammlung von Daten durch Befragung, Test und deren Auswertung. Sie muss neutral, vollständig und wiederholbar sein. Der Ist-Zustand soll an den Zielen und Anforderungen gemessen werden.

Vor jeder Evaluierung müssen Anforderungen festgestellt werden. In der Anforderungsspezifikation können folgende Kriterien eine Rolle spielen:

- Güte des Zusammenspiels mit potentiellen Testobjekten
- Möglichkeit zur Integration in die vorhandene Entwicklungsumgebung
- Plattform, auf der das Werkzeug eingesetzt werden soll
- Service und Verlässlichkeit des Herstellers
- Lizenzbedingungen, Preis, Wartungskosten

Nach [Spi10, S. 224] besteht der Auswahlprozess aus fünf Schritten:

1. Anforderungsspezifikation
2. Aufstellen einer Übersichtsliste der Kandidaten durch Recherche im Internet
3. Vorführung von Werkzeugdemos
4. Evaluierung der Werkzeuge der engeren Wahl
5. Review der Ergebnisse und Werkzeugauswahl

Die Kriterien sollen gewichtet und K.-o.-Kriterien festgelegt werden. Im Internet existieren Seiten, die Informationen über Werkzeuge in gebündelter Form anbieten und Links auf die Seiten der Werkzeuganbieter enthalten. So wird herausgefunden, welche Werkzeuge existieren und für welche Programmiersprachen und Plattformen sie welchen Funktionsumfang bieten. Kongresse zu den Themen Softwaretest, -Analyse und -Verifikation, bei denen sich Anbieter von Software-Prüfwerkzeugen vorstellen, fanden während der Erstellung der Arbeit nicht statt. Während der Evaluierung der besten Anbieter sollen vor allem folgende Punkte verifiziert werden:

- Werden Leistungsmerkmale eingehalten beziehungsweise erreicht?
- Herstellersupport

3.9 Datenmodellierung

Im Anschluss an die Evaluierung wurde ein Konzept erstellt, in dessen Rahmen eine Datenmodellierung stattfand. Im Folgenden werden Grundlagen dazu beschrieben.

Ein Modell ist auf Basis der Erkenntnis entworfen, die zu der Zeit bestand, als das Modell aufgestellt wurde. Viele Modelle werden im Laufe der Entwicklung verfeinert. Zur Entity Relationship Modellierung müssen wichtige Objekte (Entities) in einer Organisation und deren Eigenschaften (Attribute) und Beziehungen (Relationships) identifiziert werden. [Bar92, S. 7] Das Modell ist unabhängig von einem bestimmten Datenbanksystem.

Eine Entity ist eine logische Zusammenfassung von wichtigen Objekten, die in der realen oder abstrakten Welt existieren. Ein Attribut ist ein Detail, das den Zustand einer Entity näher bestimmt. Attribute können nur im Kontext einer Entity existieren. Jedes Entityexemplar kann zu einem bestimmten Zeitpunkt für jedes Attribut nur einen Wert haben. [Bar92, S. 229] In einer relationalen Datenbank ist jede Entity eine Tabelle und jedes Attribut eine Spalte. Entities müssen eindeutig identifizierbar sein, mindestens zwei, jedoch nicht mehr als acht Attribute und eine Beziehung besitzen. Zwei Entities werden mit einer Beziehung verbunden. Sie gibt an, was ein Entityexemplar mit einem anderen zu tun hat.

Die Datenmodellierung ist der logische Entwurf. Je präziser die reale Welt erfasst und im Datenmodell beschrieben wird, um so leichter ist es, entsprechende Regeln zur Wahrung der Datenintegrität zu definieren.

Grundlage des Konzepts relationaler Datenbanken ist die Relation. Im Zusammenhang mit relationalen Datenbanken wird eine Relation durch eine Tabelle beschrieben. Eine Relation besteht aus Attributen und Tupeln. [WID10] Attribute entsprechen den Spaltenköpfen, Tupel entsprechen den Zeilen einer Tabelle. Der Zusammenhang zwischen Attributen und deren Werten innerhalb einer Tabelle und die Verknüpfung von Tabellen durch Fremdschlüssel stellen Relationen dar. Eine mathematische Beschreibung des relationalen Datenmodells kann man in [Vos08] finden. In der Abbildung 3.4 sieht man die Gegenüberstellung der Grundbegriffe für eine relationale Datenbank, ein Relationen-Modell und ein Entity-Relationship-Modell.

Relationale Datenbank	Relationen-Modell	Entity-Relationship-Modell (ERM)
Wertebereich (Domäne, Domain)	Wertebereich (Domäne, Domain)	Wertebereich (Domäne, Domain)
Kopfzeile	Relationstyp/Relationsformat	Entitätstyp
Spaltenüberschrift	Attribut	Attribut
Inhalt	Relation	Entitätsmenge(-set)
Inhalt	Fremdschlüssel	Beziehung (Relationship)
Zeile	Tupel	Entität
Zeile	Attributwert	Attributwert

Abbildung 3.5: Gegenüberstellung von Grundbegriffen [WID10]

In dem Datenmodell, das während der Erarbeitung dieser Diplomarbeit erstellt wurde, wird ein Binary Large Object (BLOB) verwendet. Ein BLOB ist ein Datenfeld, in dem große Mengen binärer Daten, zum Beispiel Bilder, gespeichert werden können. Die Interpretation des Inhalts eines BLOB wird vom Anwenderprogramm übernommen.

3.10 Oracle Forms

Im Rahmen der Diplomarbeit fand eine Entwicklung mit Oracle Forms statt.

Oracle Forms ist ein Entwicklungswerkzeug, das die WYSIWYG-Erstellung und Programmierung von Datenbank-gestützten Dialogmasken erlaubt. Die Programmierung erfolgt in PL/SQL oder Java. [WIF10]

PL/SQL (Procedural Language/SQL) ist eine Programmiersprache der Firma Oracle. PL/SQL verbindet SQL mit einer prozeduralen Programmiersprache. Durch prozedurale Programmierung soll Quellcode wiederverwendbar und einfach zu gestalten sein. Ein Programm wird in kleine Teilprobleme, sogenannte Prozeduren, gegliedert. Der kleinste unteilbare Schritt ist eine Anweisung. Eine Zusammenfassung des Codes in logische Einheiten ist möglich. Unterstützt werden Variablen, Bedingungen, Schleifen und Ausnahmebehandlungen. [WIP10] Im Gegensatz zu JDBC werden die SQL-Anweisungen nicht als Zeichenketten erzeugt und an eine Datenbankschnittstelle übergeben, sondern fügen sich nahtlos in den Programmcode ein. Datenbanktrigger können programmiert werden. Ein Datenbanktrigger ist eine Funktion, die zu einem bestimmten Zeitpunkt ausgeführt wird. Er wird von einem Datenbankmanagementsystem aufgerufen, wenn Daten einer Tabelle hinzugefügt, geändert oder gelöscht werden.

Ähnlich dem Sparsamkeitsprinzip „Ockhams Rasiermesser“ gibt es das „KISS-Prinzip“. KISS ist ein Akronym. Eine mögliche Bedeutung lautet „Keep it simple and smart“ („Mach es einfach und schlau“). Es besagt, dass eine einfache, minimalistische und leicht verständliche Lösung eines Problems als am besten angesehen wird. [WIK10] Diesem Prinzip wurde während der Erstellung der Diplomarbeit Aufmerksamkeit geschenkt.

4 Anforderungsanalyse

Eine Werkzeugauswahl erfolgt nach [Spi08, S. 330] in folgenden Phasen:

1. Grundsätzliche Entscheidung über den Einsatz eines Werkzeugs
2. Identifikation der Anforderungen
3. Evaluation
4. Auswertung und Auswahl

In diesem Kapitel wird die Identifikation funktionaler und nichtfunktionaler Anforderungen an das System beschrieben. Die Anforderungen lassen sich aus den Zielen für den Werkzeugeinsatz ableiten. Es sollen die Kriterien für ein Testfallverwaltungssystem bestimmt werden, die nötig sind, um es in den Produktentwicklungsprozess von robotron*ecount zu integrieren. Die Anforderungen wurden durch die Aufgabenstellung, Analyse des Testprozesses und Befragungen des Testteams ermittelt. Anhand der Anforderungen werden im Kapitel „Evaluation“ auf dem Markt vorhandene Systeme gegenübergestellt.

4.1 Funktionale Anforderungen

Es wird kein komplettes Testmanagementsystem gefordert, sondern ein Testfallverwaltungssystem. Das heißt, die Möglichkeit, Testdurchführungen zeitlich und personell zu planen, besitzt keine Bedeutung.

Das Werkzeug soll den Testprozess der Produktentwicklung robotron*ecount unterstützen. Dafür soll das System in die vorhandene Infrastruktur integriert werden. In der nachfolgenden Abbildung ist die Infrastruktur skizziert, in die das Testfallverwaltungssystem integriert werden soll.

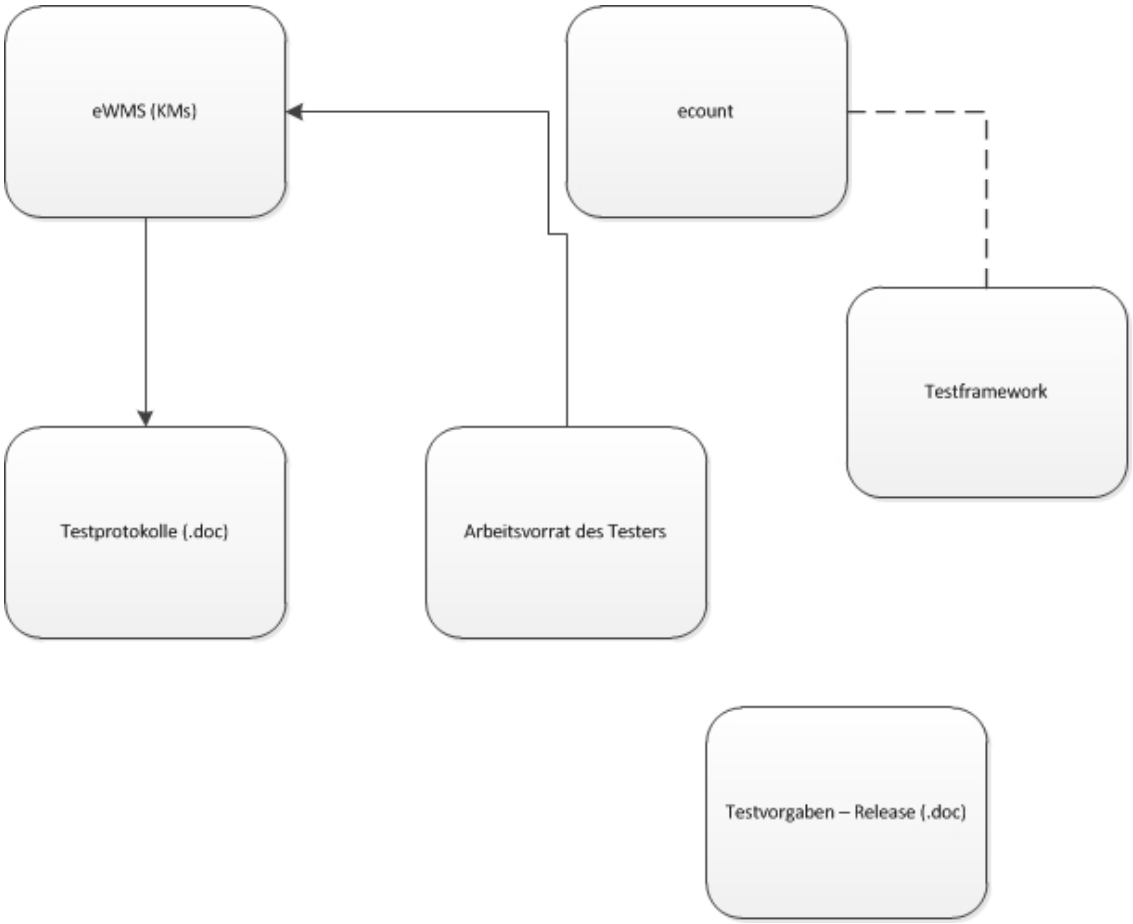


Abbildung 4.1: Vorhandene Infrastruktur

Der Arbeitsvorrat des Testers ist eine Übersicht, aus der sich der Mitarbeiter die nächste testbare Meldung heraussucht. Danach öffnet er das Fehlerverwaltungssystem robotron*eWMS, in welchem er sich die Spezifikation der Meldung ansieht, entsprechende Tests durchführt und Dokumente, zum Beispiel ein Testprotokoll, an die Meldung anhängt. Danach trägt er sich in den zu der Meldung gehörenden KM-Prozessablauf ein. Als Protokoll wird eine Microsoft Office Word-Vorlage benutzt. Unabhängig von diesem Ablauf gibt es Prüfspezifikationen, die für Releasetests verwendet werden. Das sind Microsoft Office Word-Dokumente, deren Inhalt der Definition von Testfällen (siehe dazu Kapitel 3.3) entspricht. Weiterhin gibt es ein von Robotron entwickeltes Testframework, das die automatische Ausführung von Testscripts ermöglicht. In das Testfallverwaltungssystem sollen die Testfälle des Testframeworks eingebunden werden.

Als K.-o.-Kriterien wurden folgende festgelegt:

Testfälle sollen nach Komponenten und Modulen geordnet abgelegt werden können. Bei dem Test einer Meldung sollen Testfälle, entsprechend zu bearbeiteten Komponenten und Modulen, angeboten werden.

Eine zentrale Spezifikation und Dokumentation der Tests soll über eine einheitliche Oberfläche möglich sein. Das System muss mit mehreren Benutzern gleichzeitig arbeiten können. Die Benutzeranzahl schwankt zwischen sechs und, zu Spitzenzeiten, 50.

Testprotokolle sollen strukturiert abgelegt werden und jederzeit einsehbar sein.

Über Schnittstellen soll die Integration in die vorhandene Infrastruktur möglich sein.

Weitere Kriterien sind (keine K.-o.-Kriterien):

- Dokumente sollen an Testfälle angehängt werden können
- Die Testausführung soll in einzelnen Schritten möglich sein
- Testfälle sollen verfasst, editiert und entfernt werden können
- Versionskontrolle für Testfälle
- Ressourcenkonfigurationen erstellen
- Benutzerverwaltung ⇒ Testfälle sollen zugewiesen werden
- Statistiken darstellen (Meldungen, zu denen es keine Testfälle gibt, Module / Komponenten, zu denen es keine Testfälle gibt, Einsatz von Testfällen, Intensität der ausgeführten Testfälle (wie viele Schritte je Test), durchgeführte Testfälle pro Woche)

4.2 Nicht funktionale Anforderungen

In diesem Abschnitt werden nicht funktionale Anforderungen aufgezählt.

Das System soll erweiterbar sein. Die Arbeitsumgebung wird ein firmeninternes Netzwerk sein, somit werden allgemeine Sicherheitsvorkehrungen bereits anderweitig getroffen. Im Mittel sollen Antwortzeiten von maximal drei Sekunden auftreten. Da das System zunehmend wichtiger für die Arbeit der Qualitätssicherung wird, muss es hoch verfügbar sein. Es soll mindestens zu 95% der Arbeitszeit bereitstehen. Supportleistungen des Herstellers sollten, mit angemessenen Reaktionszeiten, angeboten werden.

Die Anforderungen müssen priorisiert werden. Dazu wurde eine Prioritätenanalyse nach den unter [QHB10] beschriebenen Vorgaben durchgeführt. Eine so erstellte Präferenzmatrix wurde von fünf Personen, darunter Leiter und Tester, unabhängig voneinander ausgefüllt. Daraus ergab sich folgende Gewichtung von Kriterien, die während der Evaluation mit einbezogen wurde:

- Softwarequalität / Stabilität ⇒ 11%
- Auftretende Probleme ⇒ 10%
- Übersicht ⇒ 8%
- Handhabung ⇒ 8%

- Testfallverwaltung ⇒ 8%
- Performance ⇒ 8%
- Rechtevergabe ⇒ 7%
- Struktur (Testszenarien) ⇒ 7%
- Dokumentation ⇒ 7%
- Benutzerangaben ⇒ 6%
- Gesamtübersicht (Reporting) ⇒ 6%
- Zeitmessung ⇒ 6%
- Verwendete Datenfelder (Testfälle) ⇒ 5%
- Design / Oberfläche ⇒ 3%

Im Kapitel 5 werden auf dem Markt vorhandene Systeme unter Berücksichtigung der Anforderungen betrachtet und deren Einsatzmöglichkeit für die Produktentwicklung robotron*ecount evaluiert.

5 Evaluation der Systeme

Die im folgenden Kapitel beschriebene Evaluation wurde durchgeführt, um einen Überblick über die auf dem Markt vorhandenen Systeme zu gewinnen. Die Voraussetzungen dafür wurden mit Hilfe der Anforderungsanalyse geschaffen. Es wird zunächst die Vorgehensweise der Evaluation erläutert. Im Anschluss werden die Kandidaten vorgestellt und das Ergebnis beschrieben und begründet.

5.1 Vorgehensweise

Umfangreiche Listen im Internet boten eine Quelle für aktuelle, für die Evaluation geeignete Systeme. Hinweise von Mitarbeitern, die bereits mit Testmanagementsystemen gearbeitet haben, stellten eine weitere Quelle dar. Auch in der Literatur gab es Verweise auf Listen und Empfehlungen.

Aufgrund der großen Anzahl von Systemen musste vor der detaillierten Evaluation eine Vorauswahl getroffen werden. Dabei wurde keine definierte Kriterienliste abgearbeitet. Anhand der Beschreibungen der Anbieter und Entwickler wurden die Systeme untersucht und ihren Konkurrenten gegenübergestellt. Die Systeme mussten zum Zeitpunkt der Evaluierung einen aktiven Entwicklungsstatus besitzen und in einer aktuellen Version vorliegen, die nicht älter als ein Jahr ist. Zudem sollte ein Support für mindestens drei Jahre zu gewährleisten sein. Die Produkte mussten ausführlich dokumentiert sein. Auch Systeme, die nicht in die vorhandene Infrastruktur integriert werden können, wurden nicht weiter betrachtet. Die Systeme, die den Angaben nach die Kernkriterien und für die Vorauswahl definierte Kriterien erfüllten, wurden anhand den im Abschnitt 4 aufgestellten Anforderungen evaluiert.

Für die folgend beschriebenen Bewertungen wurde das Schulnotensystem benutzt. Die Notenvergabe beschränkt sich auf ganze Zahlen.

Nach der Aufstellung der Kriterien wurden diese mit Hilfe einer Tabellenkalkulation in einem Katalog zusammengefasst, sodass während der Evaluation die Bewertungen der einzelnen Werkzeuge verdichtet werden konnte. So konnte die Benutzbarkeit der Werkzeuge gegenübergestellt werden. Zu erwähnen ist, dass die Gegenüberstellung von Kosten und Nutzen nicht den kompletten Funktionsumfang der Werkzeuge abdeckt, sondern lediglich die in der Produktentwicklung robotron*ecount benötigten Funktionen. Die Kriterien wurden in Soll- und Kann-Kriterien unterteilt.

Zusätzlich zu der Anforderungserfüllung wurde eine Bewertungsmatrix erstellt. Im Gegensatz zur Anforderungserfüllung ist die Bewertungsmatrix eine allgemeine Bewertung

der Systeme. Die Kriterien wurden gruppiert. Es wurden für jedes Kriterium Einstufungshinweise gegeben. Es gab drei mögliche Angabearten, die sich je nach Kriterium unterschieden: Schulnoten, Zeiten und freie Kommentare. Die nicht gänzlich auszuschließende Subjektivität einer solchen Bewertung konnte nicht durch eine Erhöhung der Anzahl der Bewertenden minimiert werden, weil dafür keine Arbeitszeit reserviert werden konnte. So konnte auch kein Praxistest mit mehreren Benutzern simuliert werden, der den realen Einsatz der Systeme am ehesten entsprochen hätte. Der Schulungsbedarf, der für die einzelnen Systeme notwendig ist, konnte nicht ermittelt werden. Die anzugebenden Zeiten sollten ursprünglich die Zeit wiedergeben, die ein neuer Benutzer benötigt, um verschiedene Aufgaben abzuarbeiten. Das sollte die Bedienbarkeit des Systems messen. Da mir die Systeme beim Abarbeiten der Bewertungsmatrix bereits bekannt waren, können die gemessenen Zeiten nur als verhältnismäßiger Vergleich der Systeme interpretiert werden. Die Systeme wurden mit Hilfe eines manuellen Dummy-Testfalls evaluiert.

Falls das gewählte Werkzeug den Anforderungen nicht entsprach, wurde der zusätzliche Aufwand einer Implementierung in Personentagen abgeschätzt. Um die Lizenzkosten der Systeme gegenüberzustellen, wurde folgendes Szenario angenommen: Das System wird vorerst ein Jahr mit einer schwankenden Benutzeranzahl von 6 bis 50 Personen eingesetzt.

Im Anschluss wurden die Favoriten bezüglich ihres Datenbankschemas und Quellcodes untersucht. So wurde geprüft, welches Werkzeug eine gute Ausgangsbasis zur Weiterentwicklung beziehungsweise Einbindung von Schnittstellen bietet. Außerdem wurden Aufwand und Sollvorgaben für eine Eigenentwicklung, eingebunden in das Fehlerverwaltungssystem robotron*eWMS, abgeschätzt und anfallende Kosten verglichen.

Als Server für die Systeme diente ein Red Hat Linux. Hier wurden, soweit möglich, die Server-Anwendungen der Testmanagementsysteme installiert. Das System, das als Client benutzt wurde, war ein Windows 7 64 Bit. Es ist das zukünftige Workstation-Betriebssystem der Firma Robotron. Für Systeme, die eine Oracle Datenbank benutzen, wurde eine Oracle Datenbank 10g Express Edition 10.2.0.1 unter Red Hat Linux bereitgestellt. Die anderen Systeme boten entweder eine mitgelieferte Lösung oder nutzten die unter Red Hat Linux bereitgestellte Software Lampp (Linux Apache MySQL PHP Perl) der Version 1.7.1.

5.2 Kandidaten

In diesem Abschnitt werden die Kandidaten kurz vorgestellt und Besonderheiten genannt. Alle Testfallverwaltungssysteme sind webbasiert und setzen einen Webserver und eine Datenbank voraus.

5.2.1 HP Quality Center

Evaluiert wurde die Version 10.00. Bezogen wurde sie von [HPQ10].

HP Quality Center wurde entworfen, um für den gesamten Lebenszyklus des Qualitätssicherungsprozesses und für die Automatisierung von Softwaretests eingesetzt zu werden. Es bietet wiederholbare und standardisierte Prozesse mit der Zielsetzung, das Testen einfacher zu gestalten und zu automatisieren. Berichte lassen sich in Echtzeit erstellen.

Das System bringt einen JBoss Anwendungsserver mit und lässt sich auf eine Oracle Datenbank konfigurieren. Die Benutzeroberfläche ist in englischer Sprache gestaltet. Sie wirkt übersichtlich, schlicht und professionell. Ein integriertes Dashboard wird geboten.

Clientseitig lässt sich das HP Quality Center im Internet Explorer aufrufen. Es ist nicht vorgesehen, das System selbst anzupassen.

Die Erstellung und Ausführung von Testfällen entspricht den von Robotron gestellten Anforderungen. Testfälle lassen sich in Ordner einteilen. Eine umfassende Suchmöglichkeit ist implementiert. Während der Untersuchung traten Fehler auf, die jedoch als unerheblich eingestuft werden können. Es wurden beispielsweise nicht alle Felder vollständig dargestellt. Durch das HP Quality Center werden viele Funktionen bereitgestellt, die durch die gestellten Anforderungen nicht genutzt werden.

Zu dem Produkt werden zahlreiche Workshops und Tutorials angeboten.

In der Abbildung 5.1 sind die Testschritte eine Testfalls zu sehen, der in einem Ordner liegt.

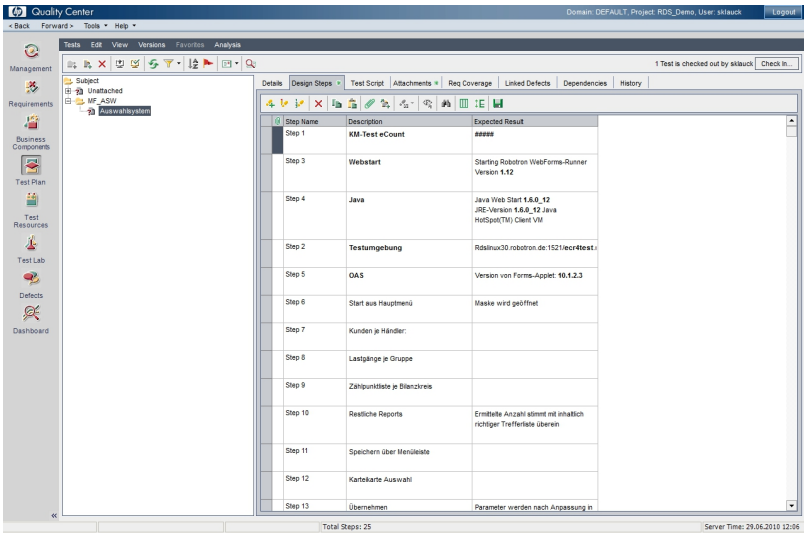


Abbildung 5.1: HP Quality Center - Screenshot

5.2.2 Oracle Application Testing Suite

Evaluiert wurde die Version 9.1.0. Das System wurde von Robotron zur Verfügung gestellt.

Wegen negativen Erfahrungen und aufgetretenen Problemen während der Untersuchung (zum Beispiel grundlos abgelaufene Sessions und Probleme bei der Installation) wurde die Software bereits während der Vorauswahl ausgeschlossen.

Das System wird hier trotzdem erwähnt, da es sich unter den Vorschlägen der Mitarbeiter befand. Deren negative Erfahrungen konnte ich jedoch bestätigen.

In der Abbildung 5.2 ist die Startseite des Werkzeugs abgebildet.

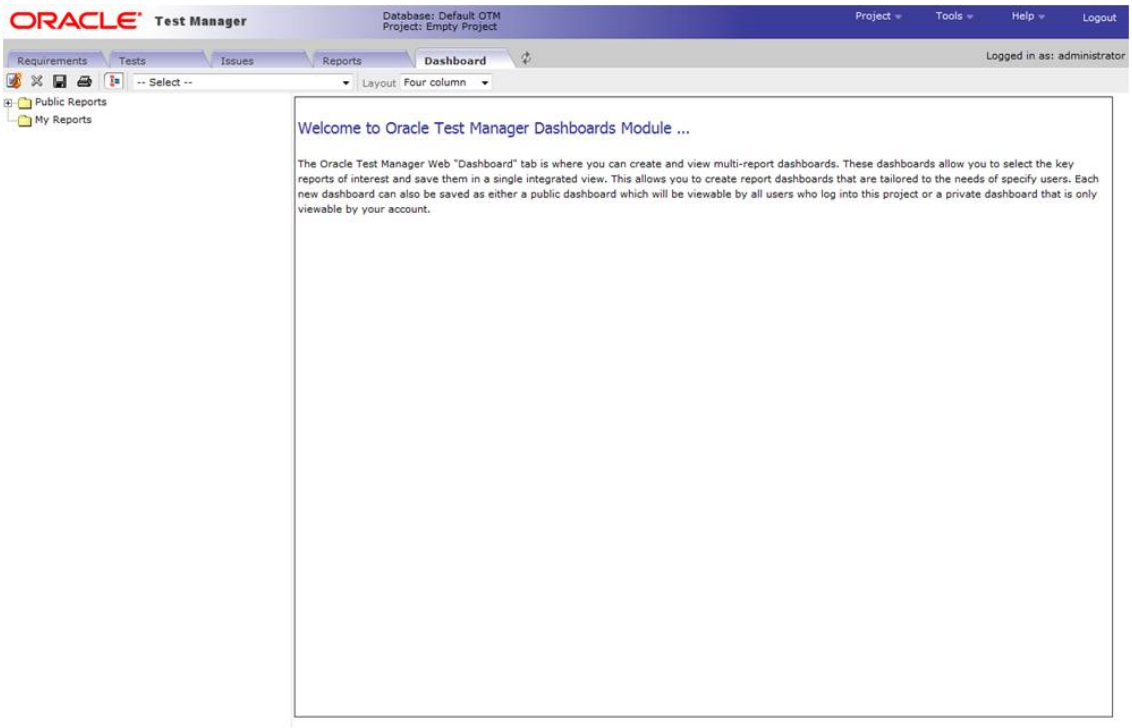


Abbildung 5.2: Oracle Application Testing Suite - Screenshot

5.2.3 IBM Rational Quality Manager

Evaluiert wurde die Version 2.0.0, die von [IBM10] bezogen wurde.

IBM Rational Quality Manager enthält Testmanagement, Qualitätsmanagement und Berichterstattung. Es können Testpläne mit Benutzerrollen, Testprozessen und Zuständigkeiten definiert werden. Das System bietet ein beliebig anpassbares Dashboard. Die Benutzeroberfläche in deutscher Sprache empfinde ich als übersichtlich und modern. Das Werkzeug reagiert schnell auf Interaktion.

Das System bringt einen Tomcat Webserver mit und lässt sich auf eine Oracle Datenbank konfigurieren.

Testfälle können den von Robotron gestellten Anforderungen entsprechend definiert und ausgeführt werden. Für Testfälle können Schablonen angelegt werden. Testfälle werden in Testsuits eingeteilt. Eine umfassende Suchfunktion ist implementiert. Außerdem können Testumgebungen angelegt werden, mit deren Hilfe Betriebssystem Images für Tests reserviert werden. Vorbildlich ist die Darstellung der Testergebnisse. Neben den positiven Eigenschaften sind während der Evaluation Fehler aufgetreten. Es wurden zum Beispiel nicht alle erstellten Testschritte ordentlich dargestellt und Registerkarten ließen sich in Einzelfällen nicht schließen.

Neue Versionen der Software werden in raschen Zyklen geboten. Das ist einerseits ein positives Anzeichen für einen aktiven Entwicklungsstand. Andererseits entstehen Risiken, wenn unternehmensspezifische Anpassungen an dem System gemacht werden. Deren Funktion könnte möglicherweise nach der Installation einer neuen Version außer Kraft gesetzt sein. IBM Rational Quality Manager ist ein umfangreiches System, das mit den von Robotron gestellten Aufgaben größtenteils unterfordert ist.

Zu dem Produkt werden Kurse angeboten.

In der Abbildung 5.3 ist das anpassbare Dashboard dargestellt, wie es nach der Installation des Produkts angezeigt wird.

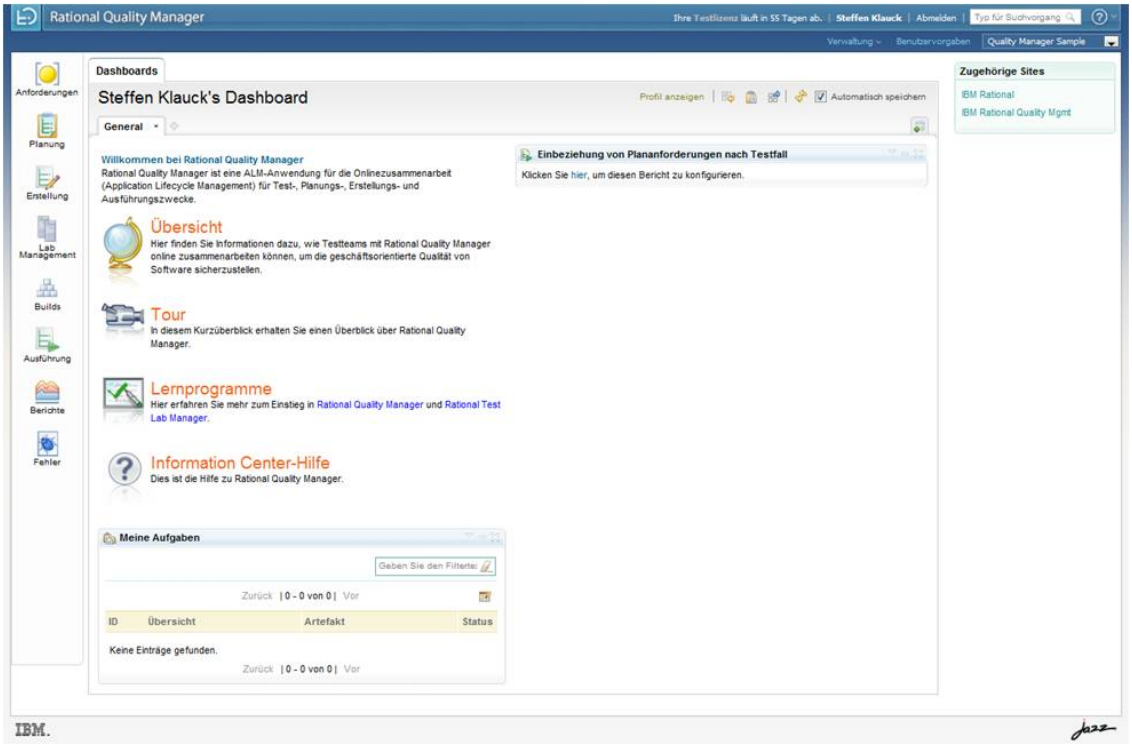


Abbildung 5.3: IBM Rational Quality Manager - Screenshot

Eine Integration der bisher vorgestellten Systeme wäre nur sinnvoll, wenn weitreichende Prozessabschnitte der Produktentwicklung robotron*ecount in die Anwendungen integriert werden. Das hätte eine fundamentale Umstellung der Prozesse zur Folge.

5.2.4 SilkCentral Test Manager

Evaluiert wurde die Version 2009, die von [SCT10] bezogen wurde. Die Version 2010 wurde zum Zeitpunkt der Evaluierung noch nicht bereitgestellt.

SilkCentral Test Manager ist eine umfassende Lösung für das Qualitätsmanagement von Anwendungen. Das Werkzeug enthält folgende Funktionsgruppen: Anforderungsmanagement, Planung, Testmanagement, Testdurchführung und Bericht. Ein Versionsverwaltungssystem kann konfiguriert werden. Mit dem SilkCentral Test Manager können Meilensteine in die Planung von Tests einbezogen werden. Das System reagiert schnell auf Benutzereingaben. Während der Untersuchung hinterließ SilkCentral Test Manager einen ausgereiften und stabilen Eindruck.

Das System lässt die Konfiguration auf eine Oracle Datenbank zu. Es wird ein mitgelieferter Ausführungsserver benötigt. Zur Evaluation wurde er unter Windows installiert. Die durchdachte Benutzeroberfläche ist in deutscher Sprache gestaltet.

Testfälle können in Container eingeordnet werden. Zur weiteren Kategorisierung können Testfällen Attribute und Parameter zugeordnet werden. Außerdem können sie nach allen Feldern von der integrierten Suche identifiziert werden. Testprotokolle können als PDF gedruckt werden. Der Hersteller nennt auf Anfrage Referenzkunden.

Es werden Tutorials zu dem Produkt angeboten.

In der Abbildung 5.4 sind die Schritte eines Testfalls dargestellt. Es werden viele Möglichkeiten geboten, die Aktionsbeschreibungen und erwarteten Ergebnisse zu formatieren.

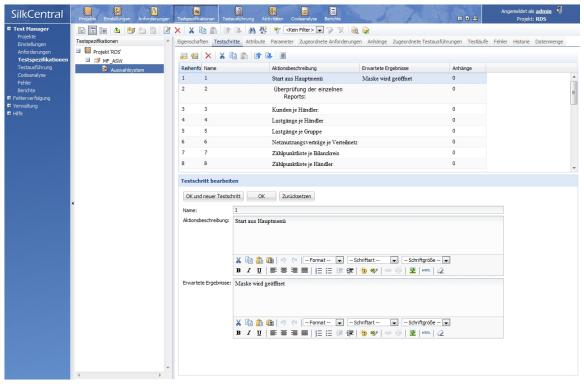


Abbildung 5.4: SilkCentral Test Manager - Screenshot

5.2.5 QaTraq

Evaluiert wurde die Version 7.0.0. Bezogen wurde sie von [QAT10].

Das Werkzeug QaTraq unterstützt den gesamten Testprozess. Der Testplan ist das zentrale Element. Testfälle werden versioniert abgelegt. Berichte können generiert werden.

Das System wurde in der Skriptsprache PHP entwickelt. Dadurch kann es weitestgehend unabhängig vom installierten Betriebssystem eingesetzt werden. Es kommt eine MySQL Datenbank zum Einsatz. Die Benutzeroberfläche ist in englischer Sprache gestaltet.

Testfälle werden in Komponenten eingeteilt. Suchmöglichkeiten sind vorhanden. Die Software bietet ansprechendes Look And Feel und hinterließ dadurch bei mir einen gereiften Eindruck. Die Produktbeschreibung hätte jedoch mehr erwarten lassen. Ich benötigte einige Eingewöhnungszeit, um das System zu bedienen. Die Übertragung der Testfälle in der Form, wie sie bei Robotron festgelegt ist, war nicht möglich.

Ein kommerzieller Support wird angeboten. QaTraq ist ein weit verbreitetes System. Der Hersteller gibt Referenzkunden an.

In der Abbildung 5.5 ist die Startseite des Systems dargestellt.

Abbildung 5.5: QaTraq - Screenshot

5.2.6 XQual XStudio

Evaluiert wurde die Version 1.3a7, die von [XQA10] bezogen wurde. Das Testmanagementsystem ist frei verfügbar.

Zur Evaluation wurde die Windows-Version der Clientanwendung installiert. Von dem System wird eine MySQL Datenbank benutzt. Die in deutscher Sprache gestaltete Oberfläche hinterließ keinen gereiften Eindruck.

Testfälle können in Kategorien und Verzeichnisse abgelegt werden. Eine Suche ist nur nach dem Namen der Testfälle möglich.

Ein kommerzieller Support wird angeboten. Der Hersteller gibt Referenzkunden an.

Nach der Installation habe ich festgestellt, dass die Angaben des Autors nicht gehalten werden. Es fehlt beispielsweise an grundlegenden Funktionen bei der Testausführung. Weiterhin ist eine geöffnete Eingabeaufforderung während der Ausführung des Systems notwendig. Das System wird nur von einer Person entwickelt. Da konkurrierende Programme bessere Möglichkeiten bieten, ist eine Verwendung für die Anforderungen nicht empfehlenswert.

In der Abbildung 5.6 ist das System nach Start des Programms dargestellt. Die Benutzerverwaltung ist geöffnet.

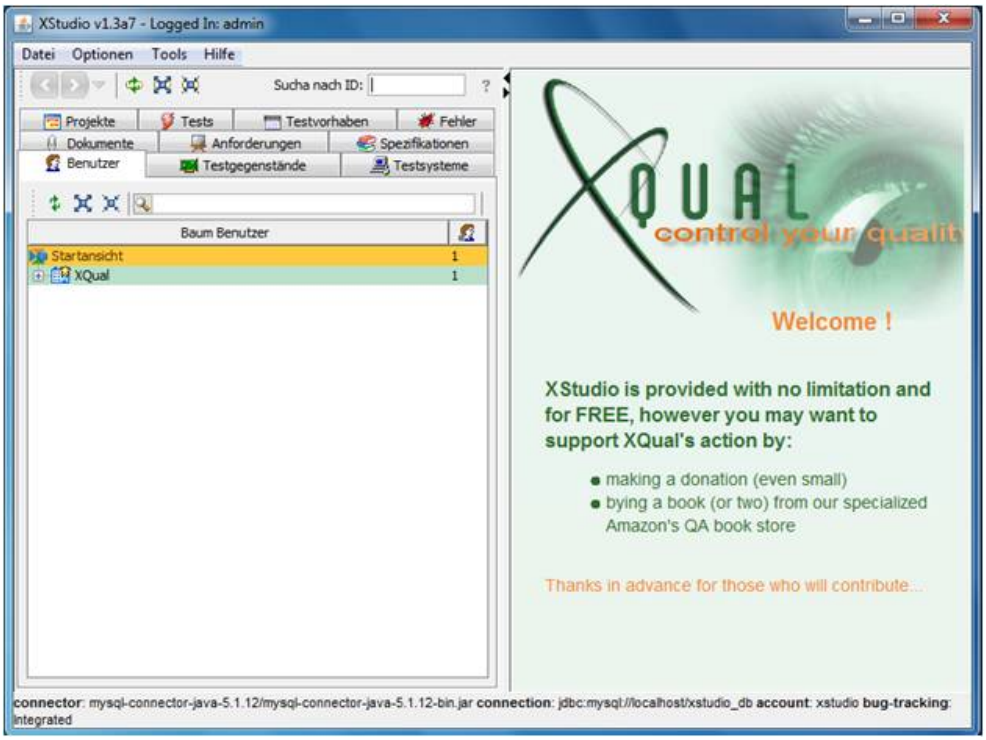


Abbildung 5.6: XQual XStudio - Screenshot

5.2.7 TestLink

Evaluiert wurde die Version 1.9 Beta4, die von [TSL10] bezogen wurde. TestLink steht unter der General Public Licence und ist frei verfügbar.

Testfälle können erstellt, verwaltet und Berichte generiert werden. Ein Testplan kann mit Builds und Meilensteinen erstellt werden. Die Software bietet gute Möglichkeiten zur Auswertung der Tests.

Das System wurde in der Skriptsprache PHP entwickelt und kann damit unabhängig vom installierten Betriebssystem eingesetzt werden. Es wird eine MySQL Datenbank benutzt. Die Oberfläche ist in deutscher Sprache gestaltet.

Testfälle können nach Testsuits und Schlüsselwörtern sortiert werden. Eine entsprechende Suchfunktion ist implementiert. Tutorials sind zu dem System vorhanden und Referenzkunden werden angegeben.

Das System ist meiner Meinung nach umständlich zu bedienen und die Zusammenhänge sind anfangs schwer zu verstehen. Es bietet jedoch vielversprechende Ansätze, sodass ich empfehle, folgende Versionen zu beobachten, zum jetzigen Zeitpunkt ist von der Verwendung abzusehen.

In der Abbildung 5.7 ist das System nach Start des Programms dargestellt.

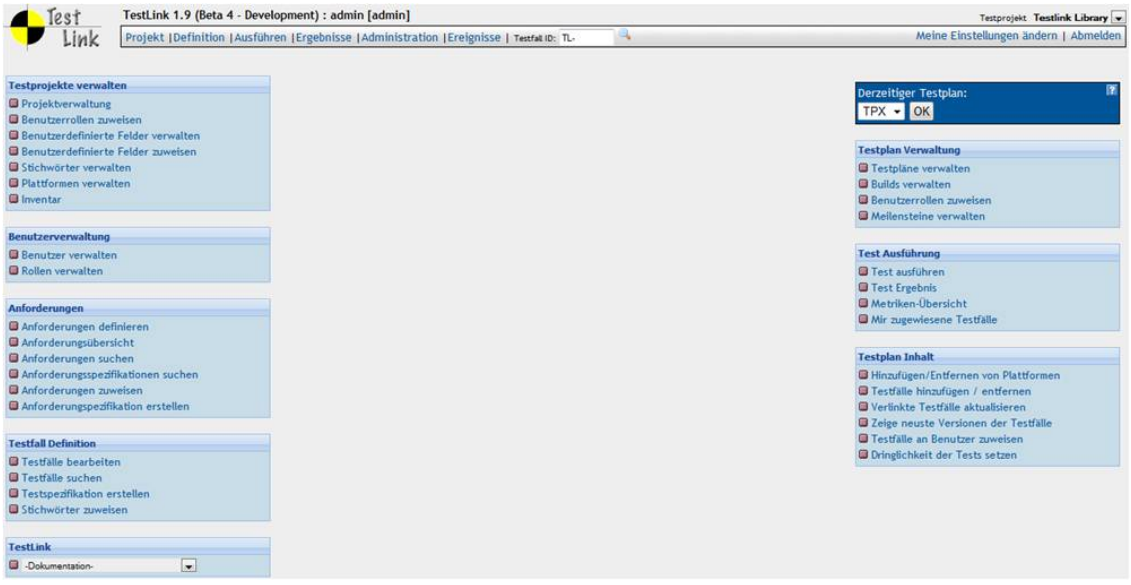


Abbildung 5.7: TestLink - Screenshot

5.2.8 Rth - Requirements and Testing Hub

Evaluiert wurde die Version 1.7.2. Bezogen wurde das Werkzeug von [RTH10].

Das System enthält die Verwaltung von Anforderungen, Testfälle können erstellt und ausgeführt werden. Ein integriertes Fehlerverwaltungssystem wird angeboten. Reportingmöglichkeiten können genutzt werden.

Der Quellcode der Software ist frei zugänglich. Rth wurde in der Skriptsprache PHP entwickelt, wodurch es unabhängig vom installierten Betriebssystem eingesetzt werden kann. Es wird eine MySQL Datenbank benutzt. Die Umstellung auf eine Oracle Datenbank ist möglich, wurde aber während der Evaluation nicht untersucht. Die Benutzeroberfläche ist in englischer Sprache gestaltet.

Testfälle werden Testsets zugeordnet. Suchmöglichkeiten werden geboten. Das System ist einfach zu bedienen und bietet eine übersichtliche Struktur. Testprotokolle können nach Microsoft Office Excel exportiert werden. Das Werkzeug überzeugte durch seine Einfachheit und die guten Integrationsmöglichkeiten. Dadurch ist eine schnelle Einarbeitung gewährleistet. Referenzkunden werden angegeben.

In der Abbildung 5.8 wird ein Testfall angezeigt. Attribute und Testschritte werden dargestellt. Mit dem Testfall können Anforderungen und Dokumente verknüpft werden.

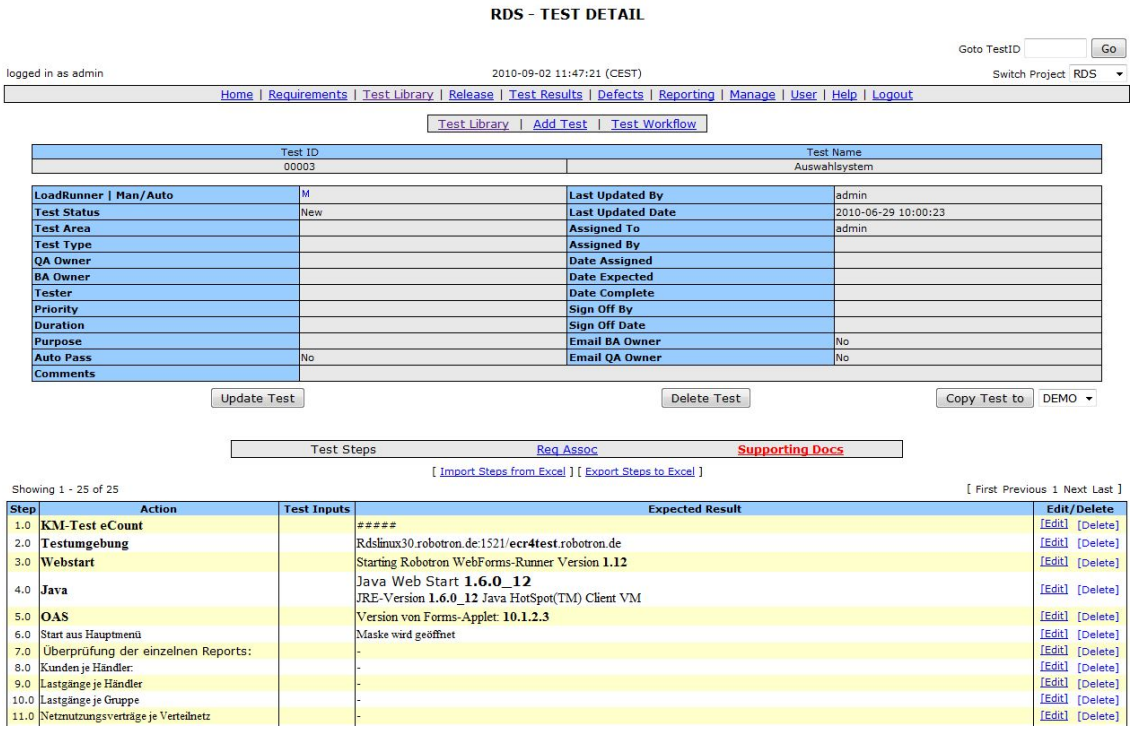


Abbildung 5.8: Rth - Screenshot

5.2.9 SalomeTMF

Das System bietet laut Aussage der Entwickler vielversprechende Ansätze, die alle Soll-Anforderungen erfüllen. Jedoch trat ein nicht definierter Fehler während der Installation auf, der, auch nach Supportanfrage, nicht behoben werden konnte.

5.3 Ergebnis

Die Entscheidung fiel zugunsten eines selbst zu entwickelnden Testfallverwaltungssystems aus. Das ist eine kostengünstige Variante. Die Entscheidung wurde von mir zunächst als Empfehlung vorgetragen und im Anschluss akzeptiert. Die These

> „Gut abgegrenzte Einzweckwerkzeuge sind wesentlich einfacher und effizienter einsetzbar als universelle Mehrzweckwerkzeuge...“ [Fru07]

stützt die Entscheidung. Im Folgenden wird die Entscheidung begründet.

Der produktive Einsatz aller Systeme, die nach der Vorauswahl betrachtet wurden, ist möglich. Bei Systemen, die keine Anpassungsmöglichkeiten bieten, wäre jedoch eine Anpassung der Prozesse nötig, was nicht den Anforderungen entspricht. Nahezu alle Systeme bieten Funktionen an, die in der Produktentwicklung robotron*ecount nicht benötigt werden. Sie müssten teilweise deaktiviert werden, um den Prozessablauf nicht zu gefährden. Die Anwendungen TestLink und XQual XStudio waren noch nicht vollständig ausgereift.

Während der Evaluation gab es nur drei Systeme, die Testschritte einzeln erstellen und ausführen ließen. Alle weiteren Anwendungen lassen nur einen Status der Durchführung der gesamten Testfälle dokumentieren. Eine Dokumentation in den Kommentaren wäre vorstellbar. Das würde aber keinen Vorteil gegenüber dem aktuellen Prozess darstellen. Weiterhin ist zu bemerken, dass Schnittstellen, die von einigen Systemen geboten werden, ausschließlich für die Integration von Fehlerverwaltung und Testautomatisierung vorgesehen sind.

Blendet man die Möglichkeit der Bearbeitung einzelner Testschritte aus, so gibt es einige vielversprechende Systeme, die in die Produktentwicklung integriert werden könnten. Abgesehen von der besseren Dokumentation und Auswertung von Tests würde der Testaufwand ansteigen.

Bei der Auswertung der Bewertungsmatrix (siehe dazu Anhang C) unter Beachtung der Prioritätenanalyse schloss QaTraq am besten ab. Vorteile gegenüber SilkCentral Test Manager und Rth sind die gut gestaltete grafische Benutzeroberfläche und die damit verbun-

dene intuitive Bedienung. So wurde am wenigsten Zeit für die Durchführung der Anwendungsfälle benötigt.

Nach der Auswertung der Bewertungsmatrix wurden folgende Kandidaten durch das Gesamtergebnis nicht weiter betrachtet:

- HP Quality Center
- XQual XStudio
- TestLink

Bei der Betrachtung von Lösungen wurde die Möglichkeit in Erwägung gezogen, ein neues System selbst zu entwickeln. Alle folgenden Bewertungen der Eigenentwicklung sind abgeschätzte Sollwerte.

Folgend ist das Ergebnis der Nutzwertanalyse dargestellt. Zur Bewertung wurde das Schulnotensystem benutzt. Niedrigere Balken stellen also eine bessere Bewertung dar.

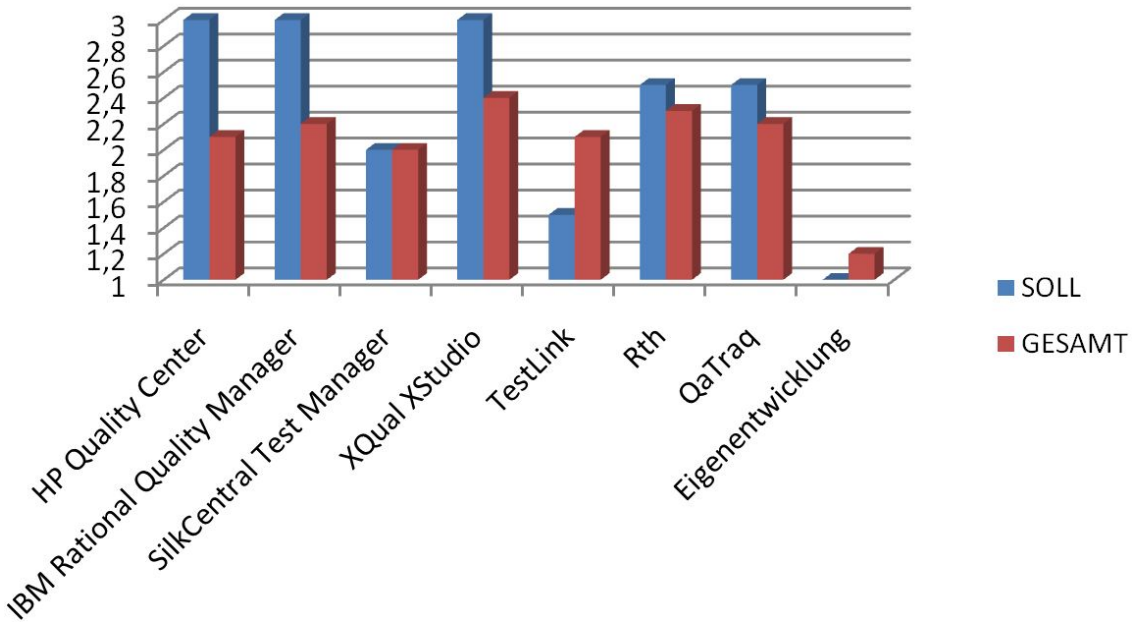


Abbildung 5.9: Ergebnis der Nutzwertanalyse

Bei den Anwendungen TestLink und SilkCentral Test Manager spricht nach dieser Darstellung nichts gegen den sofortigen Einsatz in der Produktentwicklung robotron*ecount, da die grundlegenden Anforderungen erfüllt werden. Nur kleine Unterschiede gibt es bei dem Erfüllungsgrad der erweiterten Anforderungen. TestLink wurde jedoch bereits durch die Bewertungsmatrix ausgeschlossen. Nach näherer Betrachtung der Anforderungserfüllung wurde das System IBM Rational Quality Manager ausgeschlossen. Testprotokolle konnten weder in Ordner abgelegt, noch exportiert werden. Ein weiterer Grund ist das unausgeglichene Verhältnis zwischen der Höhe der Lizenzkosten und der Anzahl der

genutzten Funktionen. Eine detaillierte Darstellung der Nutzwertanalyse ausgewählter Systeme ist folgend dargestellt.

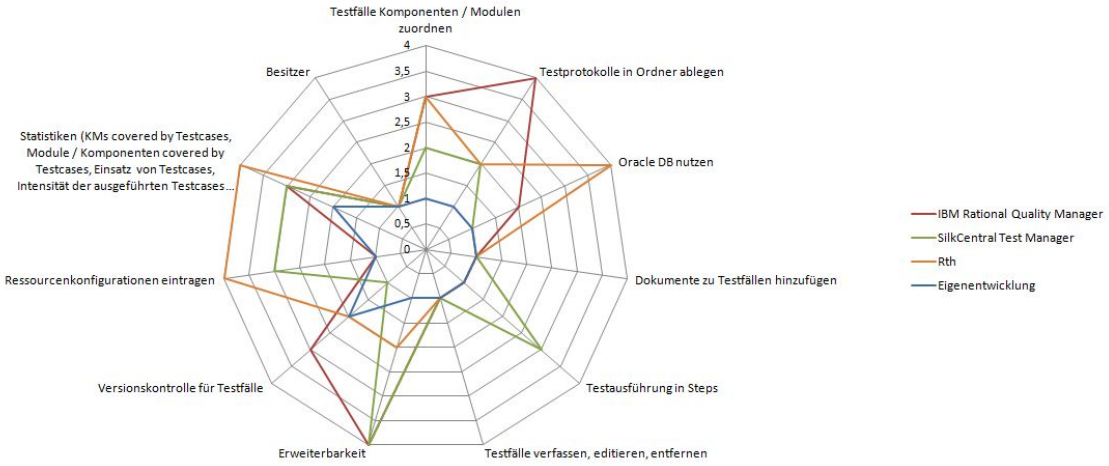


Abbildung 5.10: Nutzwertanalyse ausgewählter Systeme

Weiterhin wurde infolge der Untersuchung QaTraq ausgeschlossen. Der Grund ist, wie bereits im Abschnitt 5.2 erwähnt, dass die Testfälle, wie sie in der Produktentwicklung robotron*ecount eingesetzt werden, nicht in die Anwendung übertragen werden können.

Gegen Ende der Evaluation waren noch SilkCentral Test Manager und Rth in Konkurrenz, die die höchsten Anforderungserfüllungsgrade der übrig gebliebenen Systeme besaßen. Jedes Testmanagementsystem konnte Vorteile für sich verbuchen. Den Ausschlag für Rth gaben die geringeren Kosten, da hier keine Lizenzkosten, sondern ausschließlich Kosten für die Anpassung und Schulungskosten anfallen. Neben den Lizenzkosten für SilkCentral Test Manager würden zusätzlich Folgekosten für Support, Updates und Schulungskosten anfallen. Die im Gegenzug gebrachten Leistungen lassen keine Entscheidung für das System zu. Die Kosten ausgewählter Systeme sind folgend den Anforderungserfüllungen gegenübergestellt.

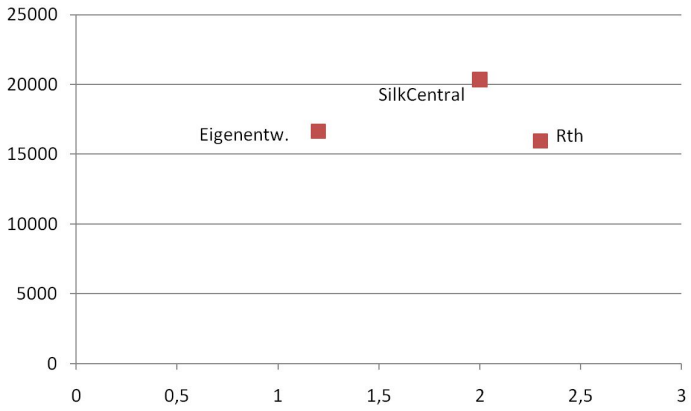


Abbildung 5.11: Gegenüberstellung von Kosten und Anforderungserfüllung

Damit verbleiben zwei mögliche Szenarien. Es kann das System Rth an die Anforderungen angepasst werden oder eine Eigenentwicklung stattfinden. Für beide Varianten wurde der Aufwand in Personentagen abgeschätzt (siehe dazu Anhang D).

Der Aufwand und die damit anfallenden Kosten unterscheiden sich nicht wesentlich. Die folgend genannten Kriterien führten zu der Entscheidung. Rth ist ein System, dass bereits seit dem Jahr 2006 existiert und von vielen Anwendern benutzt wird. Es bietet jedoch mehr Funktionen, als benötigt werden. Es besteht ein Konfliktpotential zwischen eigenen Anpassungen und Updates. Weiterhin ist der Aufwand der Weiterentwicklung schwer abschätzbar. Das System ist komplex. Allein im Hauptverzeichnis liegen 264 PHP Dateien. In der Datenbank wird mit 54 Tabellen gearbeitet. Da es zudem nur geringe Ersparnisse bei den Kosten einbringt, ist von einer Verwendung im Produktentwicklungsprozess robotron*ecount abzusehen. Demgegenüber lässt sich der Aufwand einer Eigenentwicklung, die in das Fehlerverwaltungssystem robotron*eWMS eingebunden werden kann, durch Erfahrungswerte besser einschätzen. Der Aspekt, dass von Grund auf neu entwickelt werden muss, darf nicht außer Acht gelassen werden, spiegelt sich aber in der Aufwandsabschätzung wieder. Zudem muss beachtet werden, dass bei Eigenentwicklung das System von einer Person entwickelt werden sollte, die mindestens noch lang genug beschäftigt ist, um eine andere Person einzuarbeiten. Eine Eigenentwicklung ermöglicht eine exakt angepasste und erweiterbare Lösung. Es kann entsprechend der gegebenen Infrastruktur entwickelt werden.

Diese Aspekte und die Ergebnisse aus den einzelnen Phasen der Evaluation geben den Ausschlag für die Entscheidung.

Mit dem Konzept für die Eigenentwicklung eines Testfallverwaltungssystems setzt sich das folgende Kapitel intensiv auseinander.

6 Konzept

Das in diesem Kapitel konzipierte Werkzeug wird als Testfallverwaltungssystem bezeichnet. Im Folgenden wird das Konzept für die Entwicklung des Testfallverwaltungssystems beschrieben. Die in dieser Arbeit definierten Anforderungen sollen durch Funktionen der zu erstellenden Anwendung erfüllt werden. Das Konzept ist ein grundlegender Baustein für die Entwicklung der Software. Es werden Prinzipien und Techniken festgelegt. Anschließend wird ein Datenmodell erarbeitet. Die Module der Anwendung sollen auf dem Datenmodell beruhen. In den folgenden Unterkapiteln wird die Basis für die Umsetzung geschaffen.

6.1 Prinzipien

In diesem Abschnitt werden Prinzipien festgelegt. Die prototypische Umsetzung soll den Prinzipien folgen.

Um Flexibilität der Anwendung für nachfolgende Erweiterungen zu ermöglichen, wurde beim Design der Software auf Modularität geachtet. So wird Wiederverwendbarkeit, Erweiterbarkeit und Verwaltbarkeit erreicht. Anpassungen an Veränderungen im Prozessablauf können mit einem geringen Arbeitsaufwand vorgenommen werden. Die einzelnen Module sollen funktionale, thematisch abgeschlossene und austauschbare Einheiten bilden. Sie sollen konkrete Schnittstellen besitzen. Die Software soll wenig komplex gehalten werden, um die Wartbarkeit zu erhöhen. Der Quellcode soll leserlich gehalten und kommentiert werden.

Die Realisierung der wichtigsten Funktionen ist das höchste Ziel. Erst dann folgt die Benutzbarkeit. Für die Verwaltung von Testfällen ist ein zentrales System notwendig, das von mehreren Arbeitsstationen gleichzeitig benutzt werden kann. Ziel ist es, für die Protokollierung von Testfällen und Testdurchführungen keine Microsoft Office Word-Dokumente mehr zu benutzen, damit die Anwender nicht zwischen verschiedenen Programmen wechseln müssen.

Die Anwendung und ihre Navigation sollen einfach benutzbar sein. Dazu soll beitragen, dass in allen Fenstern die gleiche Schrift, gleiche Farben, Abstände und in Position und Größe gleichbleibende Bereiche verwendet werden. Die Bereiche, die für die Entwicklung der Module vorgesehen sind, werden in der Abbildung 6.1 dargestellt. Das Testfallverwaltungssystem soll in das Fehlerverwaltungssystem robotron*eWMS integriert werden. Das Testfallverwaltungssystem soll durch die Buttonleiste am unteren Rand jedes Fensters wiedererkennbar sein. Für die Zuweisung von Schrift, Farben und Größe zu den Elementen soll auf Vererbung gesetzt werden. So wirken sich Designanpassungen

des Systems robotron*eWMS auch auf das Fehlerverwaltungssystem aus.

Abbildung 6.1: Bereiche der Fenster

Den Anforderungen entsprechend ist eine Historisierung der Testfälle und Testergebnisse nötig, um auch nicht mehr aktuelle Tests nachvollziehen zu können. Die jeweils letzte Durchführung eines Testfalls wird als aktuell definiert. Die Daten sollen zentral gespeichert werden. Alle Testergebnisse werden in das Testfallverwaltungssystem eingetragen und sollen dort zusammen mit einem Verweis auf den durchgeführten Testfall gespeichert werden. Es soll möglich sein, eine PDF-Datei mit Testergebnissen aktueller und nicht aktueller Testdurchführungen zu erstellen. Es soll auf einen Blick erkennbar sein, was wann, von wem, wie intensiv, in welcher Umgebung und mit welchem Ergebnis getestet wurde.

Auf die Abfrage der Benutzerrollen des am Fehlerverwaltungssystem robotron*eWMS angemeldeten Benutzers soll in nahezu allen Bereichen des Testfallverwaltungssystems verzichtet werden. Es wird die Ansicht vertreten, dass eine komplexe Rollenverwaltung mit Rechtevergabe die Benutzer eines Systems hemmt. Somit ist auch gewährleistet, dass Entwickler die Anwendung benutzen und zum Beispiel Testfälle erfassen können.

Lediglich die folgenden Funktionen sollen nur von Anwendern mit erweiterten Rechten benutzt werden können:

- Testumgebungen erstellen
- Schlüsselwörter, Voraussetzungen und Testschrittvorlagen bearbeiten und löschen

Zur Abfrage der erforderlichen Rechte soll das Rollensystem der Anwendung robotron*eWMS benutzt werden. Benutzer mit den Rollen Administrator und Super-Admin besitzen die benötigten Rechte.

Auf Mehrsprachigkeit wird in der Anwendung robotron*eWMS verzichtet. Deshalb wird Mehrsprachigkeit für das Testfallverwaltungssystem nicht umgesetzt. Trotzdem sollen die Bezeichner der Module keine Umlaute enthalten. Die Oberfläche soll in deutscher Sprache entwickelt werden.

Nachdem die Prinzipien für die prototypische Umsetzung festgelegt wurden, werden die zu verwendenden Techniken im nächsten Abschnitt diskutiert.

6.2 Techniken

Im Folgenden wird die Architektur der Anwendung grafisch dargestellt. Danach werden die Techniken bestimmt, die für die Entwicklung des Prototyps zu verwenden sind.

Die allgemeine Architektur der Software wird in die Komponenten Datenmodell und grafische Oberfläche aufgeteilt.

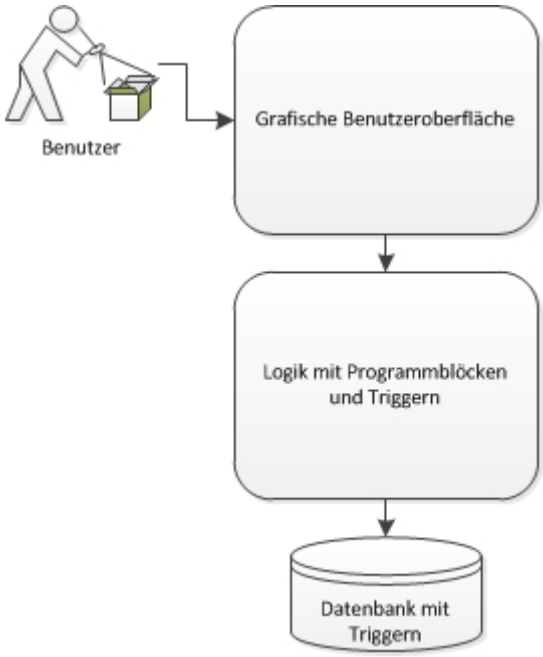


Abbildung 6.2: Komponenten der Anwendung

Für die Entwicklung soll Oracle Forms verwendet werden. Die Gründe dafür sind vielseitig. Da 70% des Fehlerverwaltungssystems robotron*eWMS in Oracle Forms entwickelt wurden, ist eine nahtlose Integration möglich. Die Entwicklungswerkzeuge stehen bei Robotron zur Verfügung. Während der prototypischen Entwicklung des Testfallverwaltungssystems kann auf das Wissen der Entwickler zurückgegriffen werden. Durch die Wahl einer dem Unternehmen vertrauten, stabilen Technik wird die Weiterentwicklung des Systems erleichtert. Die Alternative stellt Oracle ADF (Oracle Application Development Framework) dar, weil der Funktionsumfang dem von Oracle Forms ähnelt. Oracle ADF ist ein Java-Framework für die Anwendungsentwicklung. Beide Produkte, Oracle Forms und Oracle ADF, eignen sich grundsätzlich zur Erstellung der Softwarelösung. Da jedoch in absehbarer Zukunft das Fehlerverwaltungssystems robotron*eWMS weiterhin auf Oracle Forms beruht, wird auf den Einsatz von Oracle ADF verzichtet. Nachfolgend wird ein Überblick über verwendete Techniken gegeben:

- Oracle Database 10g Enterprise Edition Version 10.2.0.4.0
- PL/SQL Version 10.2.0.4.0
- Forms Version 10.1.2.3.0
- PL/SQL Editor WinMain Software Version 1.0
- Oracle SQL Developer Data Modeler Version 2.0.0

Im Unterkapitel 6.3 wird das Datenmodell konzipiert, auf dem das System zur Verwaltung von Testfällen beruhen soll.

6.3 Datenmodell

Alle Daten, die das Testfallverwaltungssystem persistent benötigt, sollen in einer Datenbank gespeichert werden. Persistente Datenhaltung bedeutet, dass die Daten über die Laufzeit der Anwendung hinweg dauerhaft gespeichert werden. Es soll eine relationale Datenbank verwendet werden. Sie soll alle Informationen zu den Testfällen und Testdurchführungen enthalten. Eine Abhängigkeit der Angaben von anderen Datenquellen soll es nicht geben. Durch die Speicherung aller Daten in einer Datenbank muss sich anwendungsseitig nicht um die Datensicherung gekümmert werden.

Für die Entwicklung des Datenmodells, dass zur Konzeption des Testfallverwaltungssystems beitrug, wurde das frei verfügbare Werkzeug Oracle SQL Developer Data Modeler eingesetzt. Die gute Abstimmung auf Oracle Datenbanken und die Abdeckung der benötigten Funktionen waren Gründe für die Entscheidung.

Für die meisten Menschen ist es einfacher, Fehler in der Arbeit anderer zu finden. Deshalb flossen Hinweise der späteren Benutzer der Software in die Entwicklung des Datenmodells mit ein. Um die Entwicklung des Datenmodells nachvollziehen zu können, wurde nach jedem Entwicklungstag eine neue Version gespeichert. Ein Datenmodell sollte nach Titel, Datum und Autor benannt sein. Das jeweilige Erstellungsdatum kann dem Dateinamen entnommen werden. Bei der Betrachtung der Datei Tm020810Sk.png sieht man also die Version des Modells vom 02.08.2010, die für die Testverwaltung von Steffen Klauck erstellt wurde.

Durch den Einsatz von Präfixen kann eine Datenbank übersichtlich gehalten werden. Je Firma und Datenbank-Produkt werden unterschiedliche Konventionen empfohlen. Die Tabellen des Testfallverwaltungssystems sollen in die Datenbank des Fehlerverwaltungssystems robotron*eWMS integriert werden. Die Tabellen des Systems robotron*eWMS sind durch das Präfix FV gekennzeichnet. FV steht für Fehlerverwaltung. Die Tabellen des Testfallverwaltungssystems sollen durch das Präfix FV_TFVS gekennzeichnet werden. TFVS steht für Testfallverwaltungssystem. Für die Namen selbst sollen keine Abkürzungen und keine Umgangssprache verwendet werden. Abkürzungen können nicht an allen Stellen vermieden werden, da andernfalls ungültige Längen für Bezeichner entstehen. Jede Tabelle soll mindestens zwei Spalten besitzen. Die in [Bar92] empfohlene Höchstanzahl von acht Attributen je Entity, und somit acht Spalten je Tabelle, ist aufgrund der Menge abzubildender Informationen nicht immer ausreichend. Die Namen der Spalten sollen in Einzahl formuliert werden und nicht den Name der zugehörigen Tabelle enthalten. Der Empfehlung, den Krähenfuß einer Vielfachheit immer links oder oben darzustellen, kann in einem umfangreichen Datenmodell nur in Ansätzen entsprochen werden. Die Tabellennamen werden in Mehrzahl formuliert.

Zu den neu zu erstellenden Tabellen des Testfallverwaltungssystems zählen zwei Haupttabellen und acht Nebentabellen, von denen fünf über K-Tabellen mit den Haupttabellen

verknüpft sind und drei ausschließlich Schlüssel-Wert Paare enthalten. Eine K-Tabelle ist eine Verbindungstabelle, die benötigt wird, um in einem relationalen Datenmodell eine n:n-Beziehung abzubilden. Weiterhin soll es 14 Tabellen geben, die für die Historisierung verwendet werden. Im Folgenden werden wichtige Tabellen beschrieben. In den dazugehörigen Abbildungen werden Ausschnitte des relationalen Datenmodells dargestellt, auf dem das Testfallverwaltungssystem beruhen soll. In den Abbildungen sind die Tabellennamen, Spalten und Indizes zu sehen. Die Buchstaben links der Spaltennamen stehen für die angegebenen Schlüssel. Ein P steht für einen Primärschlüssel, ein F für einen Fremdschlüssel und ein U für einen Unique Key. Ist ein Unique Key für eine Spalte definiert, darf jeder Wert in der Spalte einmal vorkommen. Es ist zu beachten, dass nur die Spalten durch ein F gekennzeichnet sind, deren Fremdschlüssel in der aktuellen Abbildung eine Spalte referenziert. Mit einem roten Stern sind die Spalten gekennzeichnet, die obligatorisch anzugeben sind. Hinter jeder Spalte ist der zugehörige Datentyp definiert. Die Indizes sind mit Symbolen gekennzeichnet. Indizes, die automatisch für Spalten mit Primärschlüssel angelegt wurden, sind symbolisch durch einen Schlüssel gekennzeichnet. Alle weiteren Indizes werden durch eine Raute markiert. Indizes wurden per Hand für die Spalten angelegt, denen ein Fremdschlüssel oder Unique Key zugeordnet ist. Tabellen, die für die Historisierung verwendet werden, besitzen keine Fremdschlüssel und Unique Keys. Hier wurden Indizes per Hand für Spalten angelegt, die in den zugehörigen Originaltabellen einen Primärschlüssel oder Fremdschlüssel besitzen.

Im Datenmodell vorkommende Vielfachheiten sind:

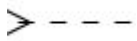


Abbildung 6.3: Zero or more

⇒Ein Objekt kann mehrfach, einfach oder nicht vorkommen.

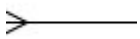


Abbildung 6.4: One or more

⇒Ein Objekt muss mehrfach oder einfach vorkommen.

Ein Überblick über das vollständige Datenmodell, das dem Testfallverwaltungssystem zugrunde liegt, ist im Anhang E abgebildet.

In der folgenden Abbildung sind die Tabellen FV_TFVS_TFAUSFUEHRUNGEN und FV_TFVS_TESTFAELLE dargestellt. Die Tabelle FV_TFVS_TESTFAELLE definiert

einen Testfall mit einem Namen und einer Beschreibung. Der Benutzer kann zusätzlich angeben, ob der Testfall einer Meldung zugeordnet sein soll. Dem Testfall wird ein Projekt und ein Benutzer des Testfallverwaltungssystems zugeordnet. Es wird eingetragen, ob der Testfall als Vorlage zur Erstellung neuer Testfälle dienen soll. Es wurde nicht als notwendig erachtet, dem Testfall eine Releaseversion des SUT zuzuordnen. Die Spalten STAND, DB_USER, HOSTNAME und OS_USER werden automatisch durch einen Datenbanktrigger gefüllt. Der Trigger wird nachträglich beschrieben. Die Spalten werden für die Dokumentation benötigt. In der Tabelle FV_TFVS_TFAUSFUEHRUNGEN werden die Ergebnisse der Testfalldurchführungen gespeichert. Es wird gespeichert, ob der Testfall bestanden wurde. Die Testumgebungen und eine getestete Releaseversion müssen angegeben werden, weil jeder Testfall in einer bestimmten Umgebung und einem bestimmten Release des SUT durchgeführt wird. Es kann ein Kommentar und eine Meldung, zu der der Testfall durchgeführt wurde, angegeben werden. Bei nicht bestandenen Testfällen kann eine Fehlerursache ausgewählt werden. Der Testfalldurchführung wird der Benutzer des Testfallverwaltungssystems zugeordnet und es wird das Projekt, in dessen Rahmen getestet wurde, eingetragen. Wenn eine Durchführung einem Testfall zugeordnet ist, wird der Testfall eingetragen. Wenn kein Testfall zugeordnet ist, wurde ein Testprotokoll unabhängig von einem Testfall angelegt. Es muss nicht zu jedem Testfall eine Testdurchführung geben.

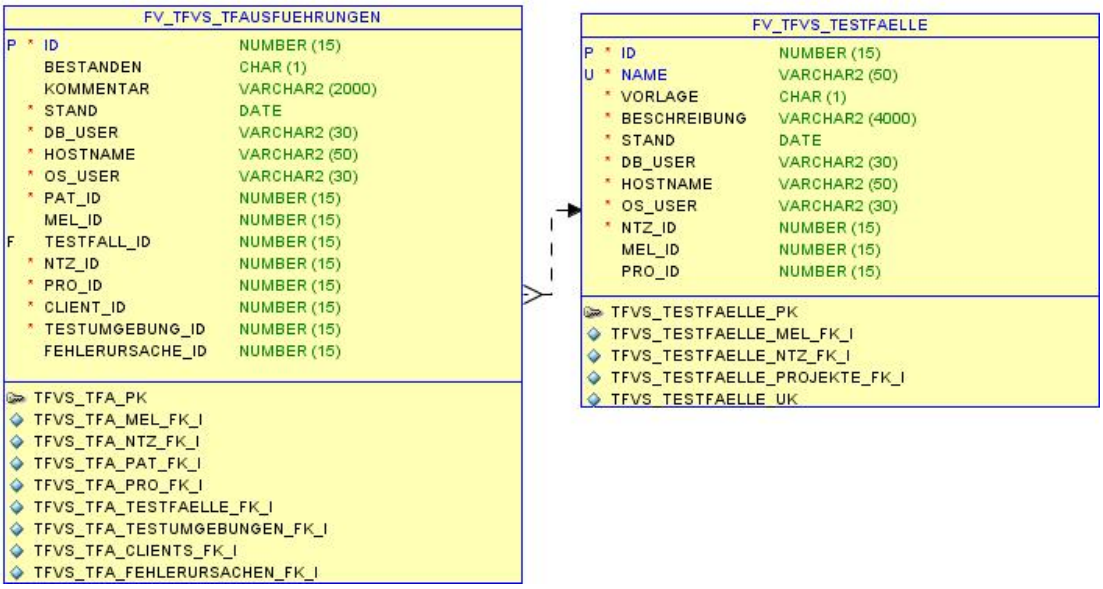


Abbildung 6.5: Die Tabellen
FV_TFVS_TFAUSFUEHRUNGEN und FV_TFVS_TESTFAELLE

Wenn ein Testprotokoll erstellt wird, das einem Testfall zugeordnet ist, sind die Testschrittdurchführungen den Schritten des Testfalls zugeordnet. In der Abbildung 6.6 sind die Tabellen FV_TFVS_TSAUSFUEHRUNGEN und FV_TFVS_TESTSCHRITTE dargestellt. Testschritte müssen durch eine Aktionsbeschreibung und ein erwartetes Ergeb-

nis beschrieben werden. Es können Vorlagen angelegt werden, um sie bei der Erstellung neuer Testschritte zu benutzen. Wird ein Testfall durchgeführt, dann wird die Durchführung der ihm zugeordneten Testschritte dokumentiert. Es kann angegeben werden, ob der Testschritt bestanden wurde. Zusätzlich kann ein Kommentar eingetragen werden.

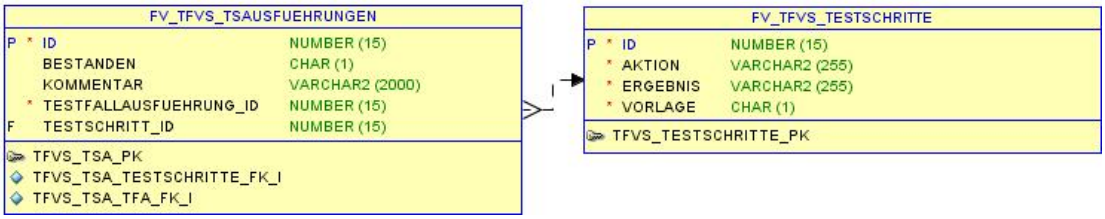


Abbildung 6.6: Die Tabellen FV_TFVS_TSAUSFUEHRUNGEN und FV_TFVS_TESTSCHRITTE

Bei der Durchführung von Testfällen muss der Benutzer eine clientseitige und eine serverseitige Testumgebung angeben, in der er getestet hat. Dafür werden die Tabellen FV_TFVS_CLIENTS und FV_TFVS_TESTUMGEBUNGEN benutzt. Zu den Angaben der clientseitigen Testumgebung zählt das Betriebssystem, die Java-Version und optional der Webbrowser. Angaben, die zu der serverseitigen Testumgebung gehören, sind die Testumgebung (gekennzeichnet durch die Datenbankverbindung), die Forms-Applet-Version (die Spalte APPLICATIONSERVER) und die WebForms Runner Version (die Spalte WEBSTART).

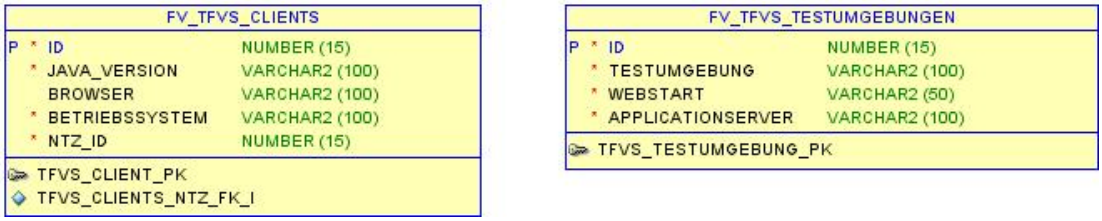


Abbildung 6.7: Die Tabellen FV_TFVS_CLIENTS und FV_TFVS_TESTUMGEBUNGEN

In der Tabelle FV_TFVS_TESTDOKUMENTE werden Dokumente in einer Spalte mit dem Datentyp BLOB gespeichert. Das Komprimieren der Dateien vor dem Speichern ist nicht notwendig, da nur JPG-Dateien, PDF-Dateien und DOCX-Dateien zulässig sein sollen. Dokumente können Testfällen, Testschritten, Testfalldurchführungen und Testschrittdurchführungen angehängt werden. Mindestens eine der Spalten TESTFALL_ID, TESTFALLAUSFUEHRUNG_ID, TESTSCHRITT_ID und TESTSCHRITTAUSFUEHRUNG_ID muss gefüllt werden. Die Prüfung übernimmt ein Datenbanktrigger.

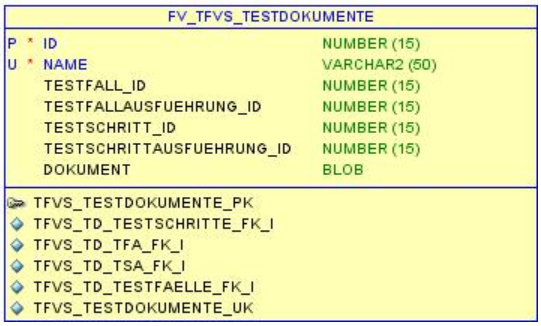


Abbildung 6.8: Die Tabelle FV_TFVS_TESTDOKUMENTE

Beispielhaft für die Tabellen

- FV_TFVS_SCHLUESSELWOERTER
- FV_TFVS_VORAUSSETZUNGEN
- FV_TFVS_FEHLERURSACHEN

ist in der Abbildung 6.9 die Tabelle FV_TFVS_SCHLUESSELWOERTER dargestellt. Die Tabelle besteht nur aus einem Schlüssel-Wert Paar. Wie der Name andeutet, wird unter Benutzung der Tabelle FV_TFVS_SCHLUESSELWOERTER eine Möglichkeit geboten, Schlüsselwörter anzulegen, die Testfällen zugeordnet werden können. So können zur Recherche nach Testfällen nicht nur die Zuordnungen zu Komponenten und Modulen herangezogen werden.

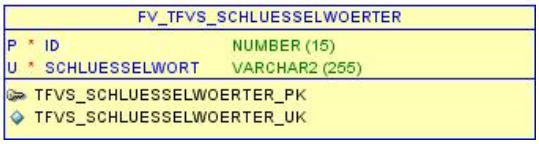


Abbildung 6.9: Die Tabelle FV_TFVS_SCHLUESSELWOERTER

Beispielhaft für die Tabellen

- FV_TFVS_K_TFKOMPONENTEN
- FV_TFVS_K_TESTFAELLEMODULE
- FV_TFVS_K_TFTESTSCHRITTE
- FV_TFVS_K_TFSCHLUESSELWOERTER
- FV_TFVS_K_TFVORAUSSETZUNGEN

ist in der folgenden Abbildung die Tabelle FV_TFVS_K_TFKOMPONENTEN dargestellt. Die Tabelle ist eine K-Tabelle. Sie ermöglicht es, einem Testfall mehrere Kom-

ponenten und mehreren Testfällen eine Komponente zuzuordnen. Zur Erstellung von K-Tabellen wurden die unter [NNB10] beschriebenen Hinweise herangezogen.

Abbildung 6.10: Die Tabelle FV_TFVS_K_TFKOMPONENTEN

Die Abbildung 6.11 zeigt die Tabelle FV_TFVS_TESTFAELLE_HISTORIE. Sie steht beispielhaft für die folgenden Tabellen:

- FV_TFVS_SW_HISTORIE
- FV_TFVS_V_HISTORIE
- FV_TFVS_TESTDOKUMENTE_HISTORIE
- FV_TFVS_CLIENTS_HISTORIE
- FV_TFVS_TESTFAELLE_HISTORIE
- FV_TFVS_TESTSCHRITTE_HISTORIE
- FV_TFVS_TFA_HISTORIE
- FV_TFVS_K_TFKOM_HISTORIE
- FV_TFVS_K_TFMOD_HISTORIE
- FV_TFVS_K_TFSW_HISTORIE
- FV_TFVS_K_TFTS_HISTORIE
- FV_TFVS_K_TFV_HISTORIE
- FV_TFVS_TSA_HISTORIE
- FV_TFVS_TU_HISTORIE

Unter Historisierung versteht man in der Informationstechnik das Speichern von nicht mehr aktuellen Daten. Dadurch wird es ermöglicht, auch im Nachhinein Situationen nachvollziehen und rekonstruieren zu können. Wird eine Datenbankzeile modifiziert, wird der zugehörige Datensatz aktualisiert. Um den Zustand vor der Modifikation nachverfolgen zu können, muss eine Historisierung der Daten umgesetzt werden. Der zeitliche Verlauf der Daten soll abgebildet werden. Historisierte Daten sollen im Rahmen dieser Arbeit lediglich abgefragt werden können. Eine Wiederherstellung der Daten ist nicht gefordert. Die Historisierung erfolgt durch Datenbanktrigger. Um die Daten übersichtlich zu speichern, werden eigene Tabellen angelegt. Die historisierten Daten bekommen eine neue ID (Identifikation) zugewiesen. Die ID ist eine Zahl, die einen Datensatz eindeutig kennzeichnet. Die ID der Originaltabellen wird in der Historisierungstabelle in

einer Spalte gespeichert, die ID_[Name der Originaltabelle] genannt wird. Die Einträge der Spalte können mehrfach vorkommen, da bei der ereignisgesteuerten Historisierung auch bei Update-Aktionen historisiert wird. Ein Insert, also das Eintragen von Daten in eine Tabelle, erzeugt in der Regel keine historisierten Daten. Der bereits erwähnte Datenbanktrigger, der die Spalten STAND, DB_USER, HOSTNAME und OS_USER füllt, wird für die Dokumentation verwendet. Er wurde für jede Historisierungstabelle angelegt. Die Spalten werden mit der sekundengenauen Systemzeit, dem an der Datenbank angemeldeten Benutzer, dem Host (Rechner, auf dem getestet wird) und dem am Host angemeldeten Benutzer gefüllt. Die Systemzeit wird benutzt, um die historisierten Daten exakt einem Testfall beziehungsweise einem Testprotokoll zuzuordnen. Werden nach dem Ende einer Testdurchführung die Ergebnisse gespeichert, so werden, wenn vorhanden, vorherige Ergebnisse in die Historisierungstabellen übertragen und können von dort wieder abgerufen werden. Die Tabelle FV_TFVS_FEHLERURSACHEN enthält Fehlerursachen. Sie können ausgewählt werden, wenn ein Testfall nicht bestanden wurde. Die Tabelle wird nicht historisiert, weil die Notwendigkeit, ein Fenster für die Pflege von Fehlerursachen zu erstellen, nicht besteht. Fehlerursachen können nur direkt in der Datenbank hinzugefügt werden.

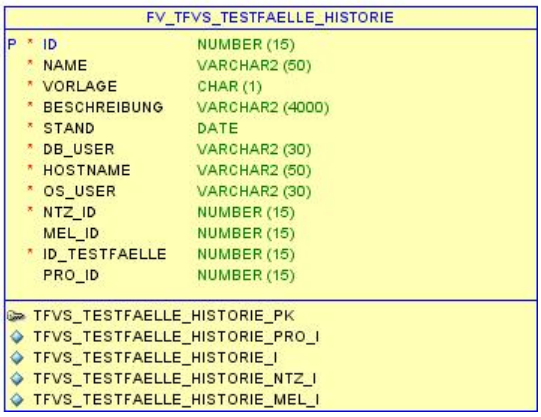


Abbildung 6.11: Die Tabelle FV_TFVS_TESTFAELLE_HISTORIE

Mit dem Datenmodell sollen die folgenden Prozesse abgebildet werden können:

1. Recherche nach vorhandenen Testfällen
2. Testfall bearbeiten oder anlegen
3. Testfall durchführen (Testergebnisse eintragen) oder Testprotokoll unabhängig von einem Testfall erstellen

Die Prozesse werden in den folgenden Abschnitten präziser beschrieben.

Recherche nach vorhandenen Testfällen

Wie bereits in den Anforderungen beschrieben, soll nach Testfällen recherchiert werden können. Testfallvorlagen sollen angezeigt werden können. Es sollen Datensätze der Tabelle FV_TFVS_TESTFAELLE abhängig der Zuordnungen in den folgenden Tabellen angezeigt werden:

- FV_TFVS_K_TFKOMPONENTEN
- FV_TFVS_K_TESTFAELLEMODULE
- FV_TFVS_K_TFSCHLUESSELWOERTER

Testfall bearbeiten oder anlegen

Beim Bearbeiten oder Anlegen eines Testfalls soll er automatisch dem Projekt zugeordnet werden, das der Benutzer im Fehlerverwaltungssystems robotron*eWMS ausgewählt hat. Dem Testfall sollen mehrere Komponenten, Module, Schlüsselwörter und Voraussetzungen zugeordnet werden. Ihm muss ein Name gegeben werden können. Die Erstellung als Vorlage und die Angabe einer Meldung soll möglich sein. Es muss eine Beschreibung des Testfalls festgelegt werden können und Testschritte sollen aus Vorlagen erstellt oder neu angelegt werden können. Hier sollen die folgenden Tabellen benutzt werden:

- FV_TFVS_TESTFAELLE
- FV_TFVS_K_TFKOMPONENTEN
- FV_TFVS_K_TESTFAELLEMODULE
- FV_TFVS_K_TFSCHLUESSELWOERTER und FV_TFVS_SCHLUESSELWOERTER
- FV_TFVS_K_TFTESTSCHRITTE und FV_TFVS_TESTSCHRITTE
- FV_TFVS_TESTDOKUMENTE

Testfall durchführen oder Testprotokoll erstellen

Wird der Testfall zu einer Meldung durchgeführt, kann sie angegeben werden. Das Projekt soll bei der Durchführung automatisch eingetragen werden, die Releaseversion des Projekts soll ausgewählt werden können. Die Auswahl von Testumgebungen muss möglich sein. Den einzelnen Testschritten soll ein Status gegeben und ein Ergebnis eingetragen werden können. Der Benutzer soll ebenso die Möglichkeit haben, die Testdurchführung mit einem Kommentar zu versehen. Hier sollen die folgenden Tabellen Anwendung finden:

- FV_TFVS_TFAUSFUEHRUNGEN
- FV_TFVS_TSAUSFUEHRUNGEN
- FV_TFVS_CLIENTS

- FV_TFVS_TESTUMGEBUNGEN
- FV_TFVS_TESTDOKUMENTE

Mit den beschriebenen Abläufen sollen im Rahmen der Arbeit folgende Anwendungsfälle abgedeckt werden:

- Recherche nach Testfällen
- Detailansicht auf Testfall
- Testfall bearbeiten
- Testfall erstellen
- Vorlagen für Testfälle und Testschritte erstellen, bearbeiten und benutzen
- Testfall durchführen
- Testdurchführung unterbrechen und zu einem späteren Zeitpunkt fortsetzen
- Testprotokoll erstellen
- Ansicht auf alte Testfalldurchführungen und alte Versionen von Testfällen
- Schlüsselwörter und Voraussetzungen verwalten
- Verwalten von Client- und Testumgebungen

Je mehr Funktionen in der Datenbank verankert werden, desto weniger muss im Nachhinein programmiert werden. So sollen an den möglichen Stellen Check Constraints verwendet werden. Ein Check Constraint ist eine benutzerdefinierte Bedingung, die ausgewertet werden muss, wenn ein Datensatz validiert wird. In den Spalten, in denen angegeben wird, ob ein Objekt eine Vorlage ist, und in den Spalten mit dem Name BESTANDEN soll jeweils mit einem Check Constraint geprüft werden, dass nur die Werte Y und N angegeben werden. Auf den Einsatz des Datentyps boolean soll verzichtet werden, um Erweiterbarkeit zu gewährleisten. So wäre zum Beispiel der Wert „Intern getestet“ für die Spalte BESTANDEN einer Testdurchführung denkbar. Weiterhin soll für jede Tabelle ein Datenbanktrigger definiert werden, der die ID mit dem nächsten Wert einer entsprechenden Sequenz füllt. Eine Sequenz generiert einen numerischen Wert, der automatisch hochgezählt wird. Auch die Historisierung und das Befüllen der Spalten STAND, DB_USER, HOSTNAME und OS_USER soll mit Datenbanktriggern umgesetzt werden.

Nachdem die Grundlage für die Speicherung von Daten geschaffen wurde, wird die Anwendung im Folgenden in Module unterteilt.

6.4 Module

Durch das Aufteilen auf mehrere Module lässt sich eine Anwendung einfacher auf Fehler prüfen, die Performance steigt und die Anwendung wird skalierbar. Es sollen 13 Formmodule entwickelt werden, die das Datenmodell benutzen. Sie werden in Haupt- und Nebenmodule unterteilt. Die Nebenmodule sollen nur aus modalen Fenstern bestehen. Ein

modales Fenster muss vom Benutzer geschlossen werden, bevor die Verarbeitung fortgesetzt werden kann. Die modalen Fenster sollen mit nur einer Schaltfläche ausgestattet werden. Wird sie ausgewählt, soll der Benutzer zurück zu einem Hauptmodul gelangen. Den Hauptmodulen werden folgende Funktionen zugeordnet:

- Recherche
- Detailansicht zu Testfällen
- Erstellen und Bearbeiten von Testfällen
- Durchführung eines definierten Testfalls
- Erstellen eines Testprotokolls unabhängig von einem Testfall

Die Hauptmodule sollen für einen Benutzer nicht parallel anwählbar sein. Die Nebenmodule sollen folgende Funktionen ermöglichen:

- Voraussetzungen für Testfälle erstellen
- Bearbeiten von Voraussetzungen von Testfällen
- Anlegen und Bearbeiten von serverseitigen Testumgebungen
- Anlegen und Bearbeiten von clientseitigen Testumgebungen
- Erstellen von Testschrittvorlagen
- Bearbeiten von Testschrittvorlagen
- Erstellen von Schlüsselwörtern
- Bearbeiten von Schlüsselwörtern

6.5 Schnittstellen

Eine Modul-Schnittstelle definiert die Beschreibung der Möglichkeiten zum Verbinden von Modulen. Module, die über eine passende Schnittstelle verfügen, können verbunden werden. Man unterscheidet gebende und nehmende Schnittstellen. Beim Entwurf komplexer Systeme beginnt man mit der Definition der Modul-Schnittstellen. [CRE10]

Im Folgenden werden die Module, beschrieben durch die zugeordneten Funktionen, und die wichtigsten vermittelnden Parameter grafisch dargestellt:

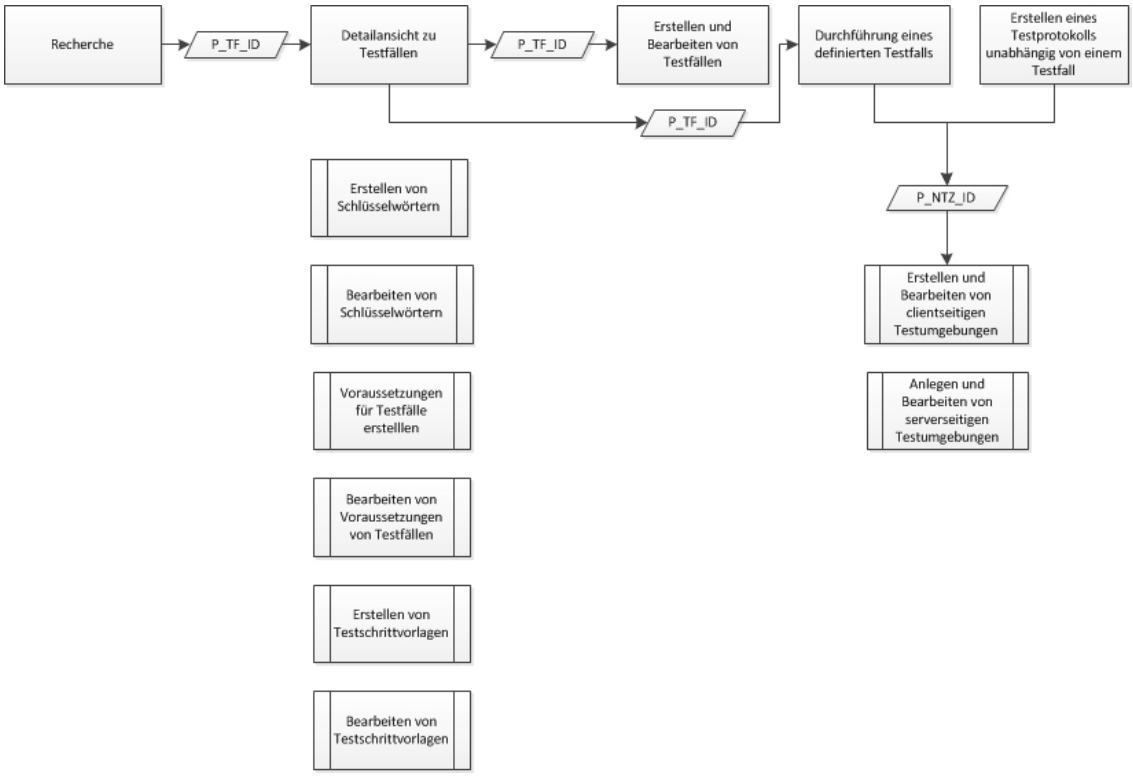


Abbildung 6.12: Module und vermittelnde Parameter

Der Parameter P_TF_ID kennzeichnet einen Testfall und der Parameter P_NTZ_ID einen Benutzer des Systems. Der Einstieg in die Anwendung wäre an vielen Stellen mit der Übergabe entsprechender Parameter denkbar.

Im nächsten Kapitel werden Aspekte der prototypischen Umsetzung des Konzepts diskutiert.

7 Prototypische Umsetzung

Dieses Kapitel beschreibt die prototypische Umsetzung der Anwendung. Zunächst wird die Benutzeroberfläche des Testfallverwaltungssystems erklärt. Anschließend folgt eine Beschreibung wichtiger Codeausschnitte.

Bei der Entwicklung wurde von Anfang an darauf geachtet, dass die in den Kapiteln 3 und 4 beschriebenen Feststellungen und Kriterien berücksichtigt werden. Ziel der prototypischen Umsetzung im Zeitraum der Diplomarbeit ist es, den wesentlichen Anteil der Funktionen herzustellen. Es soll gezeigt werden, dass ein produktiv benutzbares Testfallverwaltungssystem realisierbar ist.

Ein Prototyp ist ein funktionsfähiges, vereinfachtes Versuchsmodell, das ein Konzept illustriert. Die Umsetzbarkeit des Konzepts kann durch die Entwicklung eines Prototyps getestet und bewertet werden. Eine frühzeitige Resonanz späterer Anwender bezüglich der Eignung eines Produkts wird ermöglicht. Probleme und Änderungswünsche können schnell erkannt und mit wenig Aufwand behoben beziehungsweise eingearbeitet werden. Das Risiko einer Fehlentwicklung sinkt. Auf dem Weg zu einem Endprodukt soll es maximal drei bis vier Prototypen geben. In dieser Arbeit wird ein Programm mit Grundfunktionen entwickelt. So kann die Akzeptanz der Benutzer und die Notwendigkeit ergänzender Funktionen geprüft werden. Die Anwendung soll unter Berücksichtigung der Resonanz weiterentwickelt werden.

Die im Rahmen dieser Diplomarbeit entstandene prototypische Softwarelösung deckt die Bereiche Recherche, Testfallerstellung, Testfallbearbeitung und Testdurchführung ab. Die folgenden Aspekte wurden von Seiten des Unternehmens aufgrund der begrenzten Erstellungszeit nicht als notwendig erachtet und werden daher nicht durch den Prototyp abgedeckt:

- Abhängigkeit von Projekten für Testschrittvorlagen, Voraussetzungen der Testfälle und Schlüsselwörter
- Dokumente an Testfälle, Testschritte, Testfalldurchführungen und Testschrittdurchführungen anhängen
- Detailansichten alter Testdurchführungen und alter Versionen von Testfällen aufrufen

Die Modularität der Anwendung erlaubt es, die Oberfläche zu einem späteren Zeitpunkt auszubauen und weitere Ansichten hinzuzufügen.

Das Testfallverwaltungssystem wurde mit Oracle Forms entwickelt. Oracle Forms greift auf die Datenbank zu und erzeugt eine Ausgabe, welche Daten der datenbankgestützten Anwendung darstellt und deren Bearbeitung zulässt. Die schnelle Konstruktion brauch-

barer Benutzeroberflächen ist möglich. Die gute Abstimmung auf Oracle Datenbanken liegt in der Herkunft des Produkts Oracle Forms begründet. Die Quelldatei (*.fmb) wird in eine ausführbare Datei (*.fmx) übersetzt und vom Laufzeitmodul von Oracle Forms ausgeführt. Eine Datei stellt ein Modul dar.

In dem folgenden Unterkapitel wird die grafische Benutzeroberfläche des Prototyps beschrieben. Danach werden Codeausschnitte der entwickelten Anwendung erläutert.

7.1 Benutzeroberfläche

> „If your design makes 80 percent of the use cases easy, and the remaining 20 percent are at least possible (with a little work on the user's part), your design is doing as well as can be expected." [Tid06, S. 45]

Ziel ist es, das Testfallverwaltungssystem so aufzubauen, dass 80 % der Anwendungsfälle einfach durchführbar sind.

Im Aufbau ihrer Oberfläche orientiert sich die Anwendung an dem Fehlerverwaltungssystem robotron*eWMS. So kann mit einer gewohnten Benutzeroberfläche gearbeitet werden. In einer vertrauten Umgebung findet sich der Benutzer schneller zurecht. Die Fenster des Testfallverwaltungssystems sind in die Anwendung robotron*eWMS integriert.

Die Größe des benutzbaren Bereichs der Fenster des Fehlerverwaltungssystems robotron*eWMS wurde während der Entwicklung neu festgelegt. Die Größe des Bereichs wurde unter Beachtung der Bildschirmgrößen der Benutzer und möglicher Einstellungen ihrer Taskleiste auf 943*598 Punkte festgelegt.

Das Menü des Fehlerverwaltungssystems robotron*eWMS enthält folgende Einträge:

- Datei
  - Rücksetzen
  - Speichern
  - Drucken
  - Schließen
- Bearbeiten
  - Ausschneiden
  - Kopieren
  - Einfügen

  - Editor

- Block
  - voriger
  - nächster
  - Leeren

- Satz
  - erster
  - voriger
  - nächster
  - letzter
  - Löschen
  - Einfügen
  - Leeren
  - Doppeln

- Feld
  - voriges
  - nächstes
  - Leeren
  - Doppeln

- Anfrage
  - Filter
  - Laden
  - Wiederholen
  - Abbrechen
  - Sätze zählen

- Hilfe
  - Hilfe
  - Tastenbelegung
  - Werteliste
  - Fehler anzeigen
  - Info

An den Einträgen des Menüs wurde nicht gearbeitet. Sie werden entsprechend der Eigenschaften der Objekte, auf denen der Cursor steht, aktiviert beziehungsweise deaktiviert. Der Cursor markiert die aktuelle Bearbeitungsposition in der Anwendung.

Im Fehlerverwaltungssystem robotron*eWMS wird eine Palette mit Schaltflächen oberhalb der nicht modalen Fenster angezeigt. Die Schaltflächen sind, abhängig der Eigenschaften des Objekts, auf dem der Cursor steht, aktiviert oder deaktiviert. Die Schaltflächen haben folgende Funktionen (von links nach rechts gemäß der Abbildung 7.1):

- Speichern
- Drucken
- Schließen
- Ausschneiden
- Kopieren
- Einfügen
- Editoraufruf
- Ersten Datensatz auswählen
- Vorhergehenden Datensatz auswählen
- Nächsten Datensatz auswählen
- Letzen Datensatz auswählen
- Löschen
- Datensatz erstellen
- Filter
- Laden
- Abbrechen
- Werteliste aufrufen

Abbildung 7.1: Palette mit Schaltflächen

Bereits durch diese Schaltflächen wird es ermöglicht, dass der Benutzer jedes Fenster, durch Betätigen der erkennbar gekennzeichneten Schließen-Schaltfläche, verlassen kann. Er wird sich in keinem Fenster gefangen fühlen. Bevor ein Fenster geschlossen wird, erfolgt eine Prüfung, ob Änderungen vorgenommen wurden. Gegebenenfalls wird der Benutzer gefragt, ob die Änderungen gespeichert werden sollen. Durch die Abbrechen-Schaltfläche kann das Formmodul verlassen werden, wenn eine Abfrage verbucht wird. Ist ein modales Fenster aktiviert, so ist die Palette mit den Schaltflächen nicht erreichbar.

Der Benutzer der Fenster des Testfallverwaltungssystems soll auf einen Blick sehen, was auszufüllen ist. Das erlaubt es ihm, vorhergehende Entscheidungen noch einmal zu ändern. Außerdem soll ihm nicht vorgegeben werden, in welcher Reihenfolge die Aufgaben zu erledigen sind. Dabei muss vermieden werden, dass zu viele dargestellte Bedienelemente dem Benutzer ein Gesamtbild erscheinen lassen, das auf ihn nervös wirkt. Es werden Erklärungen in Form von Tooltips zu den auszufüllenden Feldern angezeigt. Auf Eingabehinweise, wie in der Abbildung 7.2 dargestellt, wurde aus Platzgründen verzichtet. Gegen diese Art der Darstellung von Hinweisen spricht auch, dass Anwender keine Texte lesen.

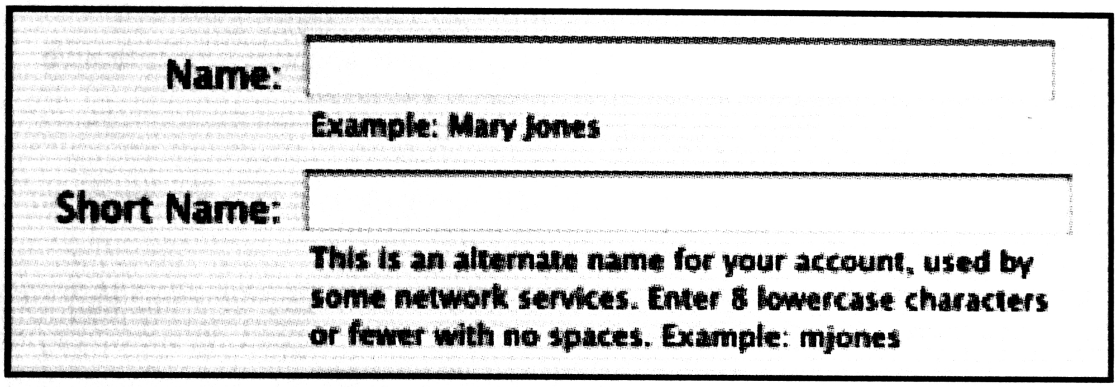


Abbildung 7.2: Eingabehinweise Mac OS X [Tid06, S. 224]

Angemessene Standardwerte wurden vergeben, um den Arbeitsaufwand des Benutzers zu reduzieren. Bei der Erstellung von Testfällen wird standardmäßig festgelegt, dass der Benutzer keine Testfallvorlage erstellen möchte. In der Werteliste der Meldungen, die Testfällen und Testdurchführungen zugeordnet werden können, werden standardmäßig nur Meldungen angezeigt, die nicht geschlossen sind und nicht den Status „kein Fehler“ besitzen. Die Einschränkung ist wegen der hohen Anzahl der Meldungen notwendig, die die Funktionen, die mit der Werteliste arbeiten, verlangsamt. Durch setzen eines Häkchens in einem Kontrollkästchen kann sich der Benutzer alle Meldungen anzeigen lassen. Die Datensatzgruppe der Werteliste wird geändert. Ist die Anzeige der Meldungen nicht eingeschränkt, steigt die Antwortzeit der Anwendung auf über drei Sekunden, was nicht den Anforderungen an das System entspricht. Weiterhin wird bei der Testdurchführung eine Releaseversion des SUT vorgeschlagen, die getestet wird. Ist einem Testfall eine Meldung zugeordnet, wird sie bei der Durchführung des Tests automatisch eingetragen. Der Status einer Testdurchführung wird standardmäßig auf „Nicht bestanden“ gesetzt. So wird nicht der Arbeitsaufwand des Benutzers reduziert, aber der Benutzer soll vor dem Speichern einer Testdurchführung nochmals zum Prüfen angeregt werden, ob der Testfall bestanden wurde.

Die Zuordnung beispielsweise mehrerer Testschritte zu einem Testfall wird in der grafischen Benutzeroberfläche mit Hilfe einer Tabelle realisiert. Die erste freie Zeile editier-

barer Tabellen kann benutzt werden, um einen neuen Datensatz anzulegen.

Damit der Benutzer die Reaktionen der Anwendung besser einschätzen kann, werden wichtige Informationen in einer Statusleiste am unteren Rand der Anwendung robotron*eWMS ausgegeben. So wird dem Benutzer zum Beispiel durch die Animation eines vertikalen Strichs signalisiert, dass eine Datenbankaktion durchgeführt wird. Weiterhin wird für jedes Objekt, auf dem der Cursor steht, angezeigt, ob ihm eine Werteliste zugeordnet ist, wie viele Datensätze für das Objekt existieren und welcher Datensatz gerade angezeigt wird.

Abbildung 7.3: Statusleiste

Von der .NET Developer Group Koblenz werden Empfehlungen für die Größe verschiedener Abstände auf einer grafischen Benutzeroberfläche gegeben. Freie Bereiche am Fensterrand sollen 12 Pixel, horizontale Abstände zwischen Schaltflächen sechs Pixel und vertikale Abstände zwischen Schaltflächen drei Pixel groß sein. Den Empfehlungen konnte nicht entsprochen werden, da versucht werden sollte, das Aussehen der Fenster möglichst präzise den vorhandenen Fenstern des Systems robotron*eWMS anzupassen.

Alle Bereiche der Fenster des Testfallverwaltungssystems, mit Ausnahme der Buttonleiste, sind durch Rahmen mit ausdrucksstarken Überschriften kenntlich gemacht. Mehr als drei Wörter werden für Überschriften nicht verwendet, damit sie einprägsam bleiben. Sie sind fettgedruckt und dunkelrot. Die Fenster sind so einfacher zu überblicken und zu verstehen. Der Inhalt ist in Abschnitte gruppiert.

Abbildung 7.4: Rahmen des Testfallverwaltungssystems

Der wichtigste Bereich der Benutzeroberfläche sollte in allen Fenstern der größte Abschnitt sein. Gemäß den Forderungen aus dem Kapitel 4 wurde ein Modul für die Recherche nach Testfällen erstellt. In der nachfolgend dargestellten Recherche der Anwendung ist die Beschreibung im Bereich „Testfälle“, gekennzeichnet durch ihre Größe, der wichtigste Bereich. Der Inhalt großer Objekte, wie die Beschreibung eines Testfalls, wird

abhängig von einer Auswahl angezeigt. Sie wird abhängig von dem aus einer Tabelle ausgewählten Testfall im gleichen Fenster dargestellt. Das Prinzip wurde beispielsweise auch auf die Kommentare der in einem modalen Fenster abgebildeten historisierten Testdurchführungen angewendet. Der Benutzer muss nicht das Fenster wechseln, um die Informationen zu sehen. Der aktuell ausgewählte Datensatz ist durch die Änderung seiner Hintergrundfarbe markiert. Testfälle können mit der Tastatur oder der Maus ausgewählt werden.

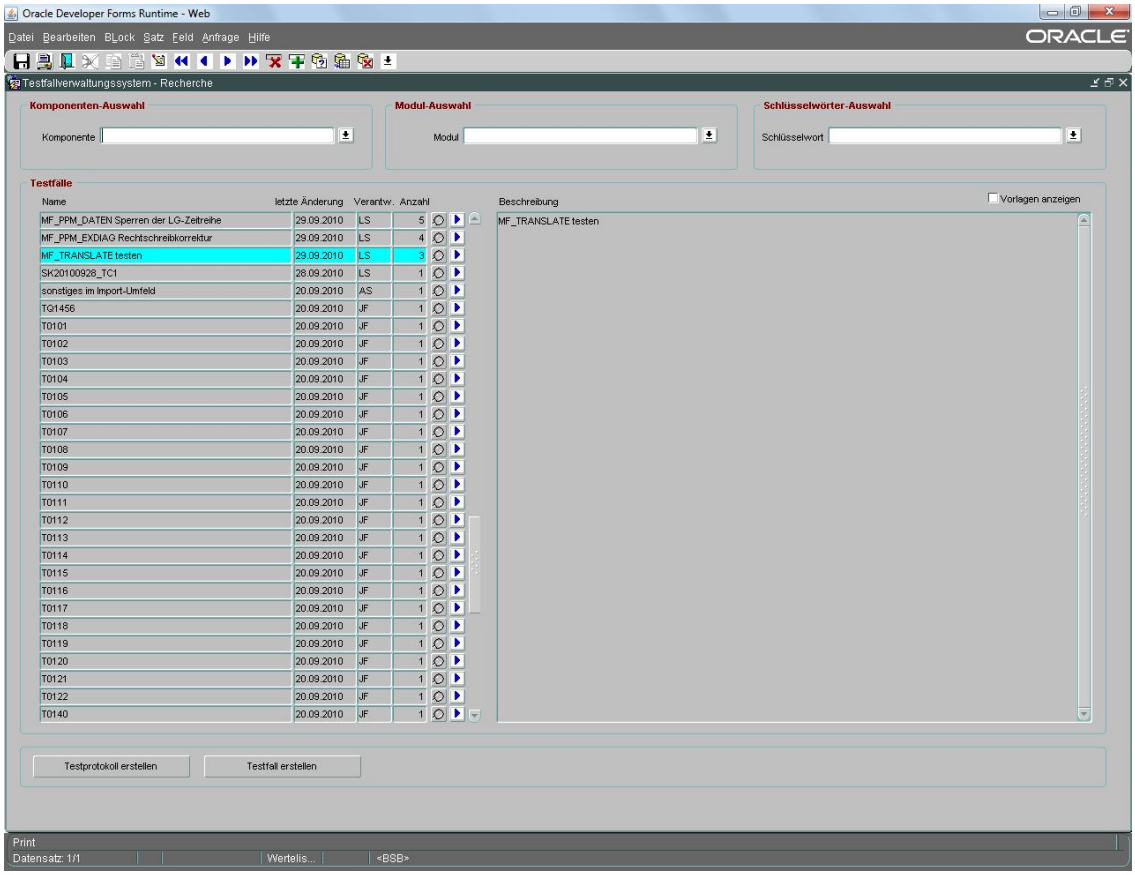


Abbildung 7.5: Recherche des Testfallverwaltungssystems

Die Textobjekte „Komponente“, „Modul“ und „Schlüsselwort“, die die Kriterien der Recherche darstellen, können mit den Werten aus Wertelisten gefüllt werden. Durch Betätigen einer Schaltfläche wird die Ausgabe der entsprechenden Werteliste in einem modalen Fenster aufgerufen.

Abbildung 7.6: Schaltfläche, die eine Werteliste aufruft

Die Angaben in den Textobjekten werden aus den Wertelisten validiert. Wenn der Be-

nutzer ein Textobjekt verlässt und einen unvollständigen Wert angegeben hat, der eindeutig einem Wert der Werteliste zugeordnet werden kann, wird der Wert der Wertliste in das Textobjekt eingetragen. Der Wert des Textobjekts wird also automatisch vervollständigt. Kann der Wert des Textobjekts nicht eindeutig einem Wert der Werteliste zugeordnet werden, wird die Werteliste geöffnet und eine mögliche Auswahl von Werten angezeigt. Die Wertelisten werden dynamisch gefüllt, indem ihnen unterschiedliche Datensatzgruppen zugewiesen werden. Sie werden ausschließlich mit den Werten gefüllt, deren Auswahl die Anzeige mindestens eines Testfalls zur Folge hat. In den Wertelisten werden Bezeichner angezeigt, die den Benutzern des Systems robotron*eWMS bekannt sind. Die Benutzer haben die Möglichkeit, in der Werteliste eine Suche durchzuführen. Die Bezeichner der Werte werden als Suchkriterien herangezogen. Wenn ein Wert ausgewählt wird, wird er in das dazugehörige Textobjekt übertragen. Nach der Auswahl eines Wertes wird die Ergebnisliste der Recherche aktualisiert. Neben den Wertelisten „Komponenten“, „Module“ und „Schlüsselwörter“ werden in dem Testfallverwaltungssystem folgende Wertelisten benutzt:

- „Meldungen“
- „Voraussetzungen“
- „Testfallvorlagen“
- „Testschrittvorlagen“
- „Releaseversionen“
- „Clientumgebungen“
- „Serverumgebungen“

In der nachfolgenden Abbildung ist die Werteliste des Textobjekts „Modul“ dargestellt.

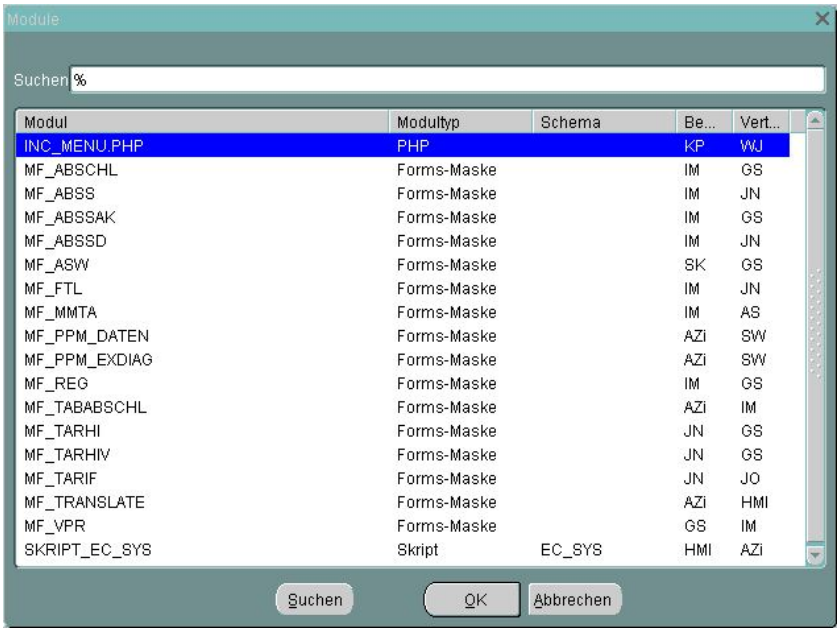


Abbildung 7.7: Werteliste des Textobjekts „Modul“

In dem Recherchefenster werden Eingangspunkte zu den anderen Fenstern des Testfallverwaltungssystems geboten. Der Benutzer kann zur Detailansicht der Testfälle, zur Testdurchführung, zur Testprotokollierung unabhängig von einem Testfall und zur Testfallerstellung gelangen. Die Bearbeitung eines Testfalls kann nicht angewählt werden. Durch die Eingangspunkte bekommt der Benutzer einen Überblick über die Funktionen der Anwendung. Die Anwendung wird dadurch selbsterklärend. Die gegebenen Bedienerhinweise sollen für Einsteiger verständlich sein. Kritisch zu betrachten ist, dass die Proportionen der Eingangspunkte nicht deren Wichtigkeit wiederspiegeln. So wird der Funktion, ein Testprotokoll unabhängig von einem Testfall zu erstellen, keine hohe Priorität zugeordnet. Dennoch ist sie durch eine Schaltfläche der Buttonleiste, die im nächsten Absatz beschrieben wird, erreichbar. Die Wichtigkeit der Schaltfläche, die zur Testdurchführung führt, wird durch ihren weißen Hintergrund betont. Das Symbol, das für die Schaltfläche verwendet wurde, wird bereits in der Palette mit Schaltflächen für die Funktion „Nächsten Datensatz auswählen“ verwendet. Trotzdem fiel die Entscheidung auf die Verwendung des Symbols, da es die Funktion der Schaltfläche am besten darstellt.

Abbildung 7.8: Schaltfläche, die zur Testdurchführung führt

Die horizontal ausgerichtete Buttonleiste am unteren Rand jedes Fensters des Testfallverwaltungssystems lässt die Navigation zu anderen Formmodulen und das Benutzen wichtiger Funktionen des aktuellen Moduls zu. Durch die Wahl der Position der Buttonleiste ist gewährleistet, dass sich die Schaltflächen am Ende des visuellen Flusses, also in der Nähe des letzten Textfelds oder Bedienelements, befinden. Durch die bereits erwähnte Festlegung der neuen Fenstergröße des Systems robotron*eWMS ist sichergestellt, dass jeder Benutzer die Buttonleiste sehen kann. Beschriftungen der Schaltflächen wurden Symbolen vorgezogen, da sie einfacher zu verstehen sind. Eine Buttonleiste bietet sich an, wenn mehr als drei bis vier Schaltflächen benutzt werden, die gruppiert werden sollen. Die Schaltflächen sind optisch gleichartig dargestellt. Nur benutzbare Elemente sind aktiviert. Im Recherchefenster wird die Schaltfläche für die Testfortsetzung ausgeblendet, wenn der angemeldete Benutzer keine Testdurchführung unterbrochen hat. Sie wird nicht deaktiviert dargestellt, weil das Unterbrechen von Testdurchführungen eine geringe Priorität besitzt und der Benutzer durch eine deaktivierte Schaltfläche nicht auf die Funktion hingewiesen werden soll. Kritisch zu betrachten ist, dass nicht alle Funktionen, die die Schaltflächen der Buttonleisten auslösen, gleichartig sind. Es gibt Schaltflächen, die das Fenster wechseln, und Schaltflächen, die Funktionen des aktuellen Fensters aufrufen. In einer Weiterentwicklung des Testfallverwaltungssystems könnten die Schaltflächen nach ihren Funktionen gruppiert werden. Ein vertikaler Trennstrich könnte die Gruppierung visualisieren. In der folgenden Abbildung ist die Buttonleiste während der Bearbeitung eines Testfalls dargestellt.

Abbildung 7.9: Buttonleiste

Nicht in allen Fenstern des Testfallverwaltungssystems ist der wichtigste Bereich durch seine Größe hervorgehoben. An den entsprechenden Stellen wird die Wichtigkeit durch den Einsatz von Farben betont. So sind editierbare Felder durch einen weißen Hintergrund und Pflichtfelder durch einen weißen Hintergrund und eine blaue Beschriftung gekennzeichnet. Weiß hebt sich gut gegenüber der Hintergrundfarbe der Fenster ab und fällt damit auf. Durch die Vorkenntnisse der Benutzer über Hervorhebungen aus dem System robotron*eWMS erkennen sie die zu bearbeitenden Felder. Bei der Testdurchführung (siehe dazu Abbildung 7.10) wird die Durchführung der Testschritte durch die Größe und Farbe des Bereichs als wichtigster Abschnitt hervorgehoben. In dem oberen Bereich „Testfall“ können folgende allgemeine Angaben zur Durchführung eines Testfalls eingetragen werden:

- Releaseversion des SUT
- Meldung, zu der getestet wurde
- Testdurchführung bestanden
- Fehlerursache, wenn die Durchführung des Testfalls nicht bestanden wurde
- clientseitige Testumgebung
- serverseitige Testumgebung

Clientseitige Testumgebungen und, abhängig von den Rechten des angemeldeten Benutzers, serverseitige Testumgebungen können erstellt, bearbeitet und gelöscht werden. In der Registerkarte „Test“ können das tatsächliche Ergebnis und der Status jedes Testschritts angegeben werden. Zusätzlich kann ein Kommentar zu der Testdurchführung eingetragen werden. Das Hinzufügen von Dokumenten ist vorbereitet. Die Standard-Druckfunktion soll bei der Weiterentwicklung des Prototyps zu einer Druckfunktion ausgebaut werden, die eine optisch wertvolle PDF-Datei mit Testergebnissen erzeugt. Das Fenster zur Erstellung eines Testprotokolls ohne Testfall ist eine eingeschränkte Variante des nachfolgend dargestellten Fensters.

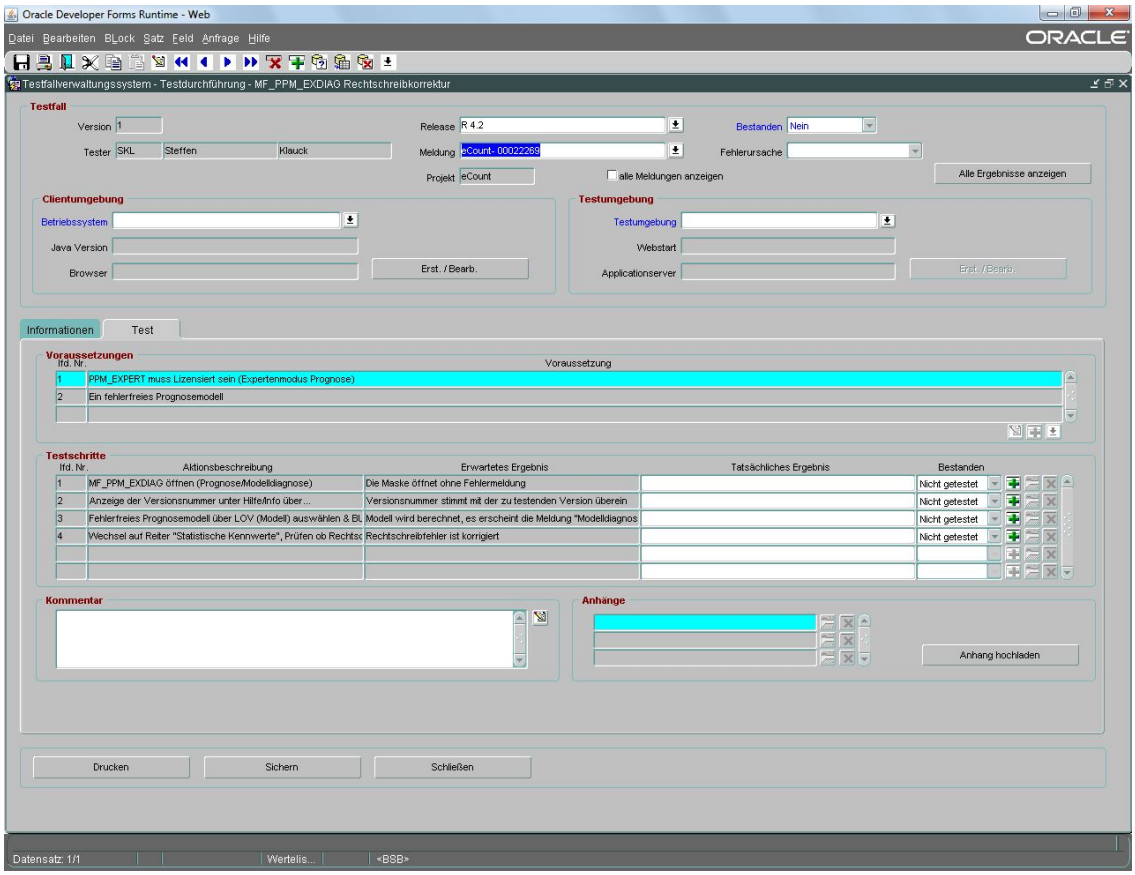


Abbildung 7.10: Testdurchführung des Testfallverwaltungssystems

In der Abbildung 7.11 ist die Detailansicht eines Testfalls dargestellt. Die Darstellung des Fensters, das im Rahmen der Diplomarbeit entstand, ist kritisch zu betrachten, da sie unübersichtlich wirkt. Es wird die Wichtigkeit keines Bereichs betont. In der Weiterentwicklung der Anwendung muss deshalb darauf geachtet werden, das Modul übersichtlicher zu gestalten. Beim Laden des Fensters werden die Informationen des Testfalls gelesen und ausgegeben. Der Name des Testfalls wird im Titel des Fensters angezeigt. Die vom Benutzer angegebenen Informationen sollten für Dritte nachvollziehbar sein. Im Folgenden werden die Bereiche des Fensters, von oben beginnend, beschrieben. Im Bereich „Eigenschaften“ sind allgemeine Informationen eines Testfalls angezeigt. Es wird zum Beispiel dargestellt, ob der Testfall eine Vorlage ist und ob er einer Meldung zugeordnet ist. In den Bereichen „Zugeordnete Komponenten“, „Zugeordnete Module“ und „Zugeordnete Schlüsselwörter“ werden die entsprechenden Zuordnungen angezeigt. Im Bereich „Beschreibung“ folgt die Beschreibung des Testfalls. Die Anzeige von Dokumenten ist im Bereich „Anhänge“ vorbereitet. Im Bereich „Voraussetzungen“ werden die Voraussetzungen des Testfalls und im Bereich „Testschritte“ die Testschritte angezeigt.

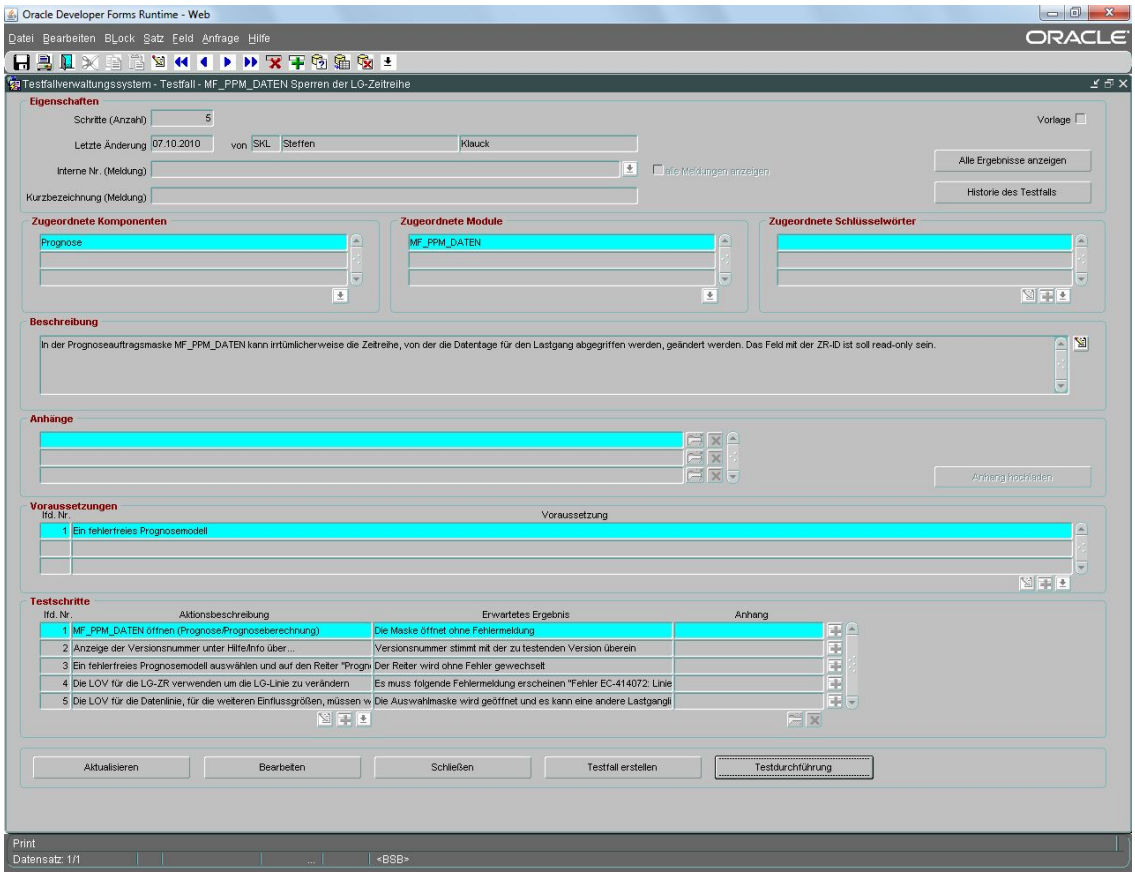


Abbildung 7.11: Detailansicht eines Testfalls des Testfallverwaltungssystems

Für die Bearbeitung und Erstellung von Testfällen wird ein Modul benutzt. In der Abbildung 7.12 ist das nicht modale Fenster des Moduls bei der Erstellung eines Testfalls abgebildet.

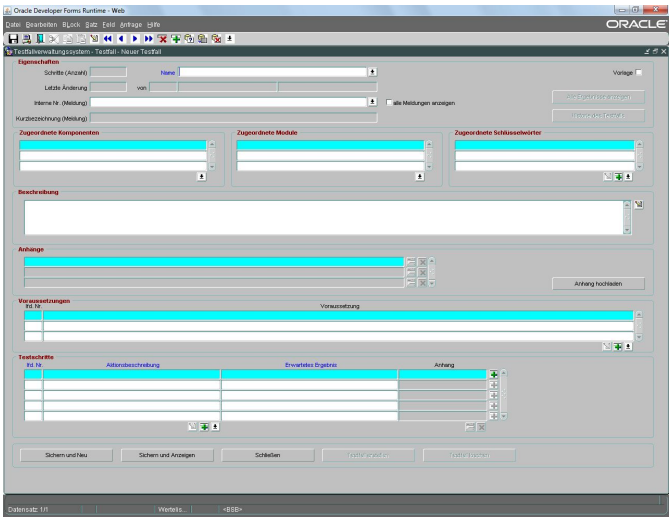


Abbildung 7.12: Erstellung eines Testfalls

Der neue Testfall kann aus einer Vorlage erstellt werden. Nach der Auswahl einer Vorlage werden die definierten Felder gefüllt. Ist man als Benutzer mit den Rollen Administrator und Super-Admin am System angemeldet, können Schlüsselwörter, Voraussetzungen und Testschrittvorlagen in modalen Fenstern nicht nur erstellt, sondern auch bearbeitet und gelöscht werden. Beim Schließen der Fenster wird dies übernommen. Soll beispielsweise ein Schlüsselwort gelöscht werden, dass einem oder mehreren Testfällen zugeordnet ist, muss die zu übernehmende Änderung bestätigt werden (siehe dazu Abbildung 7.13).

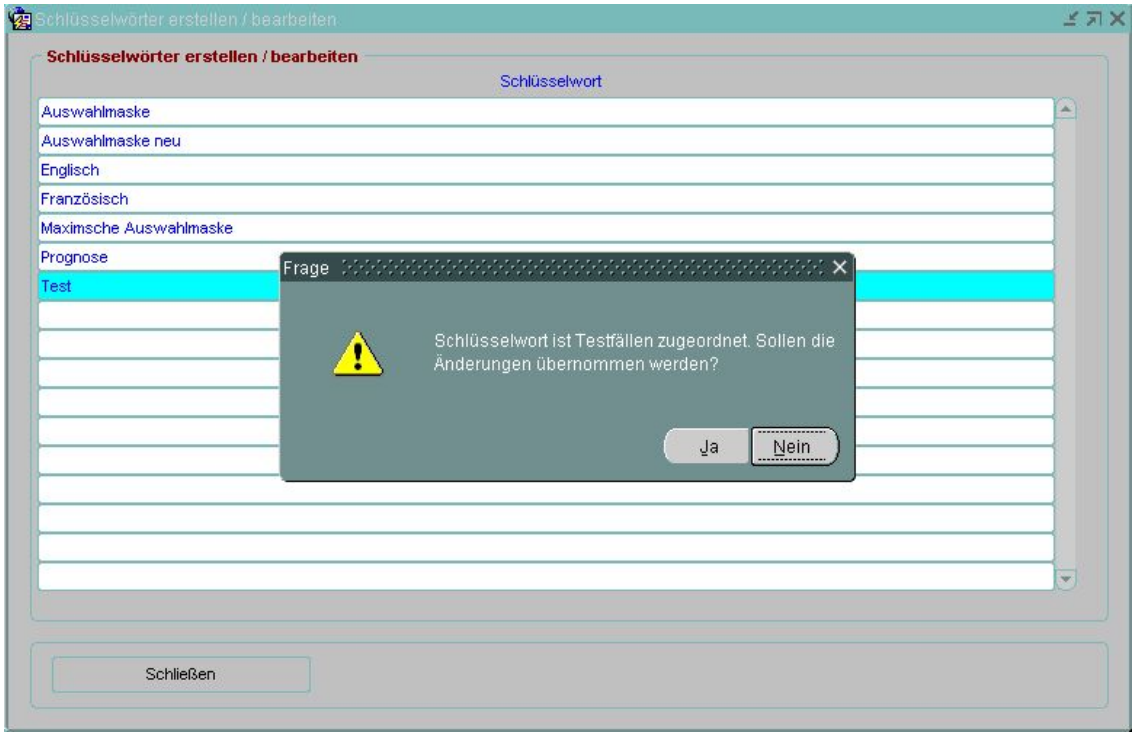


Abbildung 7.13: Schlüsselwörter bearbeiten

Wird ein vorhandener Testfall bearbeitet, so wird eine neue Version des Testfalls angelegt. Alle Daten der vorherigen Version werden in den Historisierungstabellen gespeichert.

Zusätzliche Informationen werden auf zwei verschiedene Arten angeboten: Sie sind über Schaltflächen erreichbar, die modale Fenster aufrufen, oder sie sind in einer anderen Registerkarte enthalten. Wertelisten, historisierte Testfallversionen und historisierte Testergebnisse werden in modalen Fenstern dargestellt. Bei der Testdurchführung werden weiterführende Informationen des durchgeführten Testfalls in der Registerkarte „Informationen“ dargestellt. Dazu zählen die Zuordnungen zu Komponenten, Modulen und Schlüsselwörtern, die Beschreibung des Testfalls und angehängte Dokumente. Die Benutzeroberfläche kann durch das Auslagern zusätzlicher Informationen einfacher gehalten, die Aufmerksamkeit des Benutzers konzentriert und Platz gespart werden. Der Einsatz von Bildlaufleisten für komplette Fenster der Anwendung war nicht akzeptabel.

Wichtige Codeausschnitte der Anwendung werden im nächsten Abschnitt beschrieben.

7.2 Codeausschnitte

Im weiteren Verlauf dieser Arbeit werden beispielhaft Codeausschnitte der prototypisch entwickelten Anwendung dargestellt.

Das nachfolgende Codebeispiel ist einem Programmblock entnommen, der beim ersten Aufruf des Recherchemoduls des Testfallverwaltungssystems ausgeführt wird. Der Parameter P_PRO_ID wird gesetzt. Er wird benötigt, um die Anzeige der Testfälle auf Testfälle des Projekts einzuschränken, das der Benutzer nach der Anmeldung am Fehlerverwaltungssystem robotron*eWMS ausgewählt hat. Ist kein Projekt ausgewählt, wird der Parameter auf die ID des Projekts „eCount“ gesetzt.

```
IF pkg_vars.fnc_get_var('p') IS NULL THEN
        :PARAMETER.P_PRO_ID := 11;
        pkg_vars.prc_set_var('p',:PARAMETER.P_PRO_ID);
ELSE
        :PARAMETER.P_PRO_ID := pkg_vars.fnc_get_var('p');
END IF;
```

Für die Ausgabe und Bearbeitung von Textobjekten, die lange Texte aufnehmen können, wird ein Editor zur Verfügung gestellt. Bei Betätigung der Schaltfläche, die den Editor in einem modalen Fenster aufruft, wird folgender Code ausgeführt:

```
DECLARE
  editItem VARCHAR2(100) :=
        SUBSTR(:system.trigger_item, 1,
        LENGTH(:system.trigger_item)-5);
  l_test NUMBER;
BEGIN
  GO_ITEM(editItem);
  DO_KEY('edit');
END;
```

Der Cursor wird auf das Objekt gesetzt, dessen Bezeichner dem Namen des Bezeichners der Schaltfläche, abgezogen der letzten fünf Zeichen, entspricht. Lautet der Bezeichner der Schaltfläche KOMMENTAR_EDIT, so wird der Cursor auf das Objekt KOMMENTAR gesetzt. Danach wird der Editor für das Textobjekt aufgerufen.

Die den Benutzern des Fehlerverwaltungssystems robotron*eWMS vertraute Bezeichnung von Meldungen ist die interne Nummer. Sie setzt sich aus zwei Werten zusammen, die durch den Funktionsaufruf

```
fnc_get_nr_lang(pro_id, int_f_nr)
```

verknüpft werden. Die Funktion fügt hinter den Wert „pro_id“ einen Bindestrich ein und fügt den Wert „int_f_nr“, der bis zur Fünfstelligkeit mit führenden Nullen aufgefüllt wird, an.

Die Benutzer der Anwendung sollen nicht dazu gezwungen werden, Eingaben in einer bestimmten Reihenfolge zu tätigen. Deshalb ist es nötig, Pflichtfelder nicht als erforderlich zu kennzeichnen, sondern eine Validierung erst vor dem Speichern durchzuführen. Im folgenden Beispiel wird bei dem Versuch, einen neu erstellten Testfall zu speichern, geprüft, ob ein Name angegeben wurde. Wenn kein Name angegeben wurde, wird ein Warnfenster mit der Meldung „Bitte Name angeben“ angezeigt.

```
DECLARE alert_button NUMBER;
BEGIN
IF :V_TFVS_TESTFAELLE.NAME IS NULL THEN
        SET_ALERT_PROPERTY('a_caution', ALERT_MESSAGE_TEXT,
        'Bitte Name angeben' );
        alert_button := SHOW_ALERT('a_caution');
        RAISE FROM_TRIGGER_FAILURE;
END IF;
END;
```

In Tabellen wird es ermöglicht, vorhandene Datensätze in eine neue Zeile zu kopieren. So können beispielsweise schnell ähnliche Testumgebungen erstellt werden. Dabei muss beachtet werden, dass jeder Datensatz durch eine ID gekennzeichnet ist, die nicht kopiert werden darf. Durch das nachfolgende Codebeispiel wird das Kopieren der ID im modalen Fenster zur Erstellung clientseitiger Testumgebungen verhindert:

```
DUPLICATE_RECORD;
:FV_TFVS_CLIENTS.ID := NULL;
SYNCHRONIZE;
GO_ITEM('fv_tfvs_clients.betriebssystem');
```

Beim Speichern soll die Anzeige der Anzahl geänderter Datensätze verhindert werden. Dazu muss die Systemvariable MESSAGE_LEVEL vor jeder Übergabe der Datensätze angepasst werden. Sie speichert die Fehlergrenzstufen 0, 5, 10, 15 und 20. Der Ausgangswert ist 0. Oracle Forms unterdrückt alle Meldungen mit einer Fehlergrenzstufe, die gleich der angegebenen Fehlergrenzstufe oder niedriger als die angegebene Fehlergrenzstufe ist. Die Übergabe von Datensätzen wird durch den folgenden Code realisiert:

```
:SYSTEM.MESSAGE_LEVEL := '20';
COMMIT;
:SYSTEM.MESSAGE_LEVEL := '0';
```

Beispielhaft für die Behandlung eines Fehlers steht der folgende Codeausschnitt. In dem Recherchefenster der Anwendung wird geprüft, ob der aktuell angemeldete Benutzer eine Testdurchführung unterbrochen hat. Wenn das der Fall ist, wird die Schaltfläche „pb_tfa_fortsetzen“ sichtbar gemacht. Um zu ermitteln, ob eine Testdurchführung unterbrochen wurde, wird eine Datenbankabfrage durchgeführt. Liefert sie keine Ergebnisse,

tritt der Fehler NO_DATA_FOUND auf und die Ausführung der Prozedur wird abgebrochen. Er wird mit der Fehlerbehandlung abgefangen. Dadurch kann präzise auf bestimmte Fehlerarten reagiert werden. Das vereinfacht die Behandlung festgelegter Fehler.

```
PROCEDURE wfma_when_new_form_instance IS
BEGIN
        :PARAMETER.P_USERNAME := GET_APPLICATION_PROPERTY(USERNAME);
        SELECT id INTO :PARAMETER.P_TFA_ID
        FROM FV_TFVS_TFAUSFUEHRUNGEN
        WHERE bestanden IS NULL
        AND ntz_id = (SELECT ntz.id FROM FV_NUTZER ntz
        WHERE ntz.db_login = :PARAMETER.P_USERNAME);
EXCEPTION
WHEN NO_DATA_FOUND THEN
        SET_ITEM_PROPERTY('b_control.pb_tfa_fortsetzen',VISIBLE,
        PROPERTY_FALSE);
END;
```

Kann die Standardfunktion von Oracle Forms zum Ausgeben von vorhandenen Daten in einer Tabelle nicht benutzt werden, muss ein Cursor zum Füllen der Tabelle benutzt werden. Ein Cursor ist eine Datenstruktur im Arbeitsspeicher, die für die Abarbeitung von SQL-Befehlen benötigt wird. Für Datenbankabfragen, die mehrere Zeilen liefern, muss ein expliziter Cursor verwendet werden. Der Deklarationsabschnitt des Cursors wird mit einem Namen definiert.

```
PROCEDURE prc_query_vorlage IS
  CURSOR cur_kom IS
    SELECT tfkom.kom_id, kom.name
    FROM FV_TFVS_K_TFKOMPONENTEN tfkom, FV_KOMPONENTEN kom
    WHERE kom.id IN (SELECT kom_id FROM FV_TFVS_K_TFKOMPONENTEN)
    AND tfkom.kom_id = kom.id AND testfall_id = :PARAMETER.P_TF_ID;
BEGIN
        GO_BLOCK('v_tfvs_komponenten');
  FOR r IN cur_kom LOOP
    IF :V_TFVS_KOMPONENTEN.NAME IS NOT NULL THEN
        CREATE_RECORD;
    END IF;
    :V_TFVS_KOMPONENTEN.NAME := r.name;
    :V_TFVS_KOMPONENTEN.KOM_ID := r.kom_id;
  END LOOP;
  FIRST_RECORD;
END;
```

Die Datenbankabfrage des dargestellten Cursors liefert alle Komponenten, die einem Testfall, der durch den Parameter P_TF_ID bestimmt ist, zugeordnet sind. In der im Fenster der Anwendung dargestellten Tabelle des Datenblocks V_TFVS_KOMPONENTEN wird ein neuer Datensatz angelegt, der mit einem Ergebnis der Abfrage gefüllt wird. Der Vorgang wird wiederholt, bis der Cursor keine weiteren Datensätze liefert.

Im Folgenden wird auf die Historisierung am Beispiel der Testdurchführung eingegangen. Die Ergebnisse der Testdurchführung werden gespeichert und die Ergebnisse der vorhergehenden Durchführung des Testfalls, sofern vorhanden, historisiert. Die alte Testdurchführung wird von einem Datenbanktrigger in die Historisierungstabelle geschrieben und anschließend aus der Originaltabelle gelöscht. Danach kann die aktuelle Testdurchführung gespeichert werden. Es werden folgende Codezeilen eines Datenbanktriggers ausgeführt, bevor ein neuer Datensatz in die Tabelle FV_TFVS_TFAUSFUEHRUNGEN eingefügt wird:

```
INSERT INTO FV_TFVS_TFA_HISTORIE
(id_tfausfuehrungen, bestanden, kommentar, stand, db_user, hostname,
os_user, pat_id, mel_id, testfall_id, ntz_id, pro_id, client_id,
testumgebung_id, fehlerursache_id)
SELECT id, bestanden, kommentar, stand, db_user, hostname, os_user,
pat_id, mel_id, testfall_id, ntz_id, pro_id, client_id,
testumgebung_id, fehlerursache_id
FROM FV_TFVS_TFAUSFUEHRUNGEN
WHERE :NEW.TESTFALL_ID = testfall_id;
DELETE FROM FV_TFVS_TFAUSFUEHRUNGEN
WHERE :NEW.TESTFALL_ID = testfall_id;
```

Alle Vorgänge zum Löschen und Erstellen von Datensätzen müssen so gestaltet werden, dass es nicht zu Fehlern bei den Integritätsprüfungen kommt. Bestimmte Reihenfolgen müssen beachtet werden. Die topologische Sortierung beschreibt eine Sortierung von Elementen mit vorgegebenen Abhängigkeiten. Nach dem Ansatz, solange Elemente ohne Vorgänger zu entfernen, bis keine mehr übrig sind, kann das Löschen von Elementen nach topologischer Sortierung vorgenommen werden. Im übertragenen Sinn müssen beim Löschen von Datensätzen aus dem in dieser Diplomarbeit erstellten Datenmodell zuerst Datensätze aus Tabellen gelöscht werden, auf die keine Pfeilspitze zeigt. So müssen beispielsweise vor dem Löschen eines Testfalls die Zuordnungen aus den folgenden Tabellen gelöscht werden:

- FV_TFVS_K_TFKOMPONENTEN
- FV_TFVS_K_TESTFAELLEMODULE
- FV_TFVS_K_TFSCHLUESSELWOERTER
- FV_TFVS_K_TFTESTSCHRITTE
- FV_TFVS_K_TFVORAUSSETZUNGEN
- FV_TFVS_TFAUSFUEHRUNGEN
- FV_TFVS_TESTDOKUMENTE

Erläuterungen zur topologischen Sortierung findet man unter [WIO10].

In diesem Kapitel wurde die im Rahmen der Diplomarbeit entwickelte Softwarelösung ausführlich vorgestellt. Im nächsten Kapitel werden der Verlauf und die Ergebnisse der Arbeit zusammengefasst. Abschließend wird ein Ausblick gegeben.

8 Zusammenfassung

Ziel der Arbeit war es, einen Prototyp eines Systems zur Verwaltung von Testfällen in die Produktentwicklung robotron*ecount zu integrieren. Das Ziel wurde erreicht. Die speziellen Anforderungen des Unternehmens mussten berücksichtigt werden. Zunächst wurden theoretische Grundlagen betrachtet. Testen wurde als eine Tätigkeit definiert, bei der ein Programm ausgeführt wird, um Fehler zu finden. Durch das Testen kann die Qualität einer Software bestimmt werden. Im Anschluss an die Darstellung theoretischer Grundlagen wurden die Anforderungen an die benötigte Anwendung analysiert und definiert. Die Ausgangssituation wurde beschrieben. Das zu implementierende Werkzeug soll eine zentrale Verwaltung von Tests von Funktionen des Energiedatenmanagement-Systems robotron*ecount ermöglichen. Um eine den gestellten Anforderungen entsprechende Anwendung zu finden, wurden die auf dem Markt vorhandenen Systeme evaluiert. Die Möglichkeit des Einsatzes verschiedener Softwarelösungen wurde bewertet. Eine Gewichtung der Bewertungskriterien wurde in die Ermittlung des Evaluierungsergebnisses einbezogen. Im nächsten Schritt wurde für die ausgewählten Werkzeuge, die nicht den Anforderungen entsprachen, der Aufwand einer Weiterentwicklung beziehungsweise der Einbindung von Schnittstellen abgeschätzt. Um Lizenzkosten in der Evaluierung nicht zu vernachlässigen, wurde von 50 Benutzern und einer Nutzungsdauer von einem Jahr ausgegangen. Die so anfallenden Kosten wurden einer Abschätzung der Kosten einer Eigenentwicklung, eingebunden in das Fehlerverwaltungssystem robotron*eWMS, gegenübergestellt. Es wurde entschieden, dass die Eigenentwicklung einer Anwendung stattfinden soll. Begründet wurde die Entscheidung durch die Ergebnisse der Evaluierung und die Tatsache, dass bei einer Eigenentwicklung keine Lizenzkosten anfallen. Es konnte ein System entstehen, das speziell auf die Produktentwicklung robotron*ecount abgestimmt ist. Anschließend an die Evaluierung wurde ein Konzept erstellt, das die Grundlage für die Entwicklung eines Systems zur Verwaltung von Testfällen bildet. Das entworfene Datenmodell wurde beschrieben. Die Aufgabenstellung sah weiterhin vor, dass eine prototypische Umsetzung des Konzepts erfolgen muss. Viele Funktionen sind bereits in dem entwickelten Prototyp zufriedenstellend implementiert. In kürzester Zeit wurden fast alle Anforderungen erfüllt. Durch die Implementierung des Systems in die Anwendung robotron*eWMS ist es in die vorhandenen Infrastruktur der Produktentwicklung integriert. Mit der entstandenen Anwendung können Testfälle verwaltet werden. Die Software bietet dem Anwender ein Werkzeug, mit dem er nach Testfällen recherchieren, sie erstellen, bearbeiten und löschen kann. In den Rechercheergebnissen werden die Testfälle des von Robotron entwickelten Testframeworks mit angezeigt. Vorlagen für Testfälle können definiert und für die Erstellung neuer Testfälle verwendet werden. Der Benutzer wird bei der Protokollierung der Durchführung von Testfällen und einzelnen Testschritten unterstützt. Gemäß der Definition eines Testprotokolls aus dem Abschnitt 3.1 wird festgehalten, was wann, von wem, wie intensiv, in welcher Umgebung und mit welchem Ergebnis getestet wurde. Eine Historisierung von Testfällen und Testdurchführungen ist

umgesetzt. Die Anwendung ermöglicht die Verwaltung von Testumgebungen. Durch die Einfachheit des Systems wird dem Benutzer viel Zeit erspart. Die nichtfunktionalen Anforderungen konnten, mit Ausnahme der Antwortzeit der Abfrage aller Meldungen, berücksichtigt werden. Die Modularität erlaubt es, die Funktionen der Anwendung mit wenig Aufwand zu erweitern oder an Prozessänderungen anzupassen. Außerdem wird übersichtlicher Quellcode geboten. Erste praktische Tests der Anwendung durch die späteren Benutzer haben gezeigt, welche weiteren Funktionen die Benutzung der Software vereinfachen würden.

Mit einer Weiterentwicklung des Prototyps können in kurzer Zeit alle Anforderungen erfüllt werden. Der produktive Einsatz der Anwendung ist möglich. Der in der Abbildung 5.10 dargestellten Abschätzung der Nutzwertanalyse einer Eigenentwicklung kann nach der Weiterentwicklung des Prototyps ohne Einschränkung entsprochen werden.

Abschließend werden Ideen für Funktionserweiterungen des Testfallverwaltungssystems beschrieben. Es wird als sinnvoll erachtet, dass diskutiert wird, welche Erweiterungen notwendig sind. Die Recherche der Anwendung könnte um die Recherche nach Testfällen abhängig zugeordneter Meldungen erweitert werden. Zusätzlich bietet es sich an, die Ergebnisliste um die Anzeige des letzten Testdurchführungsdatums und des Ergebnisses der letzten Testdurchführung zu ergänzen. Die Historisierung der Daten des Systems könnte optimiert werden. Es wäre möglich, ein Modul anzulegen, das nur die geänderten Spalten in die Historisierungstabellen schreibt. Weiterhin könnte die Anwendung um die Möglichkeit erweitert werden, gelöschte Testfälle anzuzeigen und, wenn notwendig, wiederherzustellen. Die Projektunabhängigkeit der Anwendung soll in Zukunft hergestellt werden. Dazu muss die Projektabhängigkeit von Testschrittvorlagen, Voraussetzungen und Schlüsselwörtern realisiert werden. Das Datenmodell müsste hierfür nur geringfügig ergänzt werden. Die Einsatzmöglichkeiten der Anwendung in anderen Projekten können jedoch nicht abgeschätzt werden, da alle Untersuchungen dieser Diplomarbeit im Rahmen des Projekts „eCount“ stattfanden. Eine Zuordnung zu Komponenten und Modulen für Schlüsselwörter, Voraussetzungen und Testschrittvorlagen ist außerdem planbar. Die Anzeige kann von den Zuordnungen abhängig gemacht werden. Die laufenden Nummern der Voraussetzungen und Testschritte könnten automatisch eingetragen werden. Es wird empfohlen, für die Bearbeitung von Testumgebungen Angaben vorzugeben. Es müssten dazu alle bei Robotron vorhandenen Umgebungen gepflegt werden. Die in der Datenbank gespeicherten Testergebnisse könnten benutzt werden, um Tests zu analysieren. Es ist möglich, über den Stand der Tests zu informieren. Zeitraubende Auswertungen könnten schnell vorgenommen werden. Die Anzahl bestandener Testfälle in Verhältnis zur Anzahl nicht bestandener Testfälle zu setzen, ist eine weitere Möglichkeit. Weiterhin könnte man zum Beispiel darstellen, zu wie vielen Meldungen, Komponenten und Modulen Testfälle existieren, wie viele Testfälle je Woche durchgeführt werden oder wie viele Testschritte im Durchschnitt je Testfall abgearbeitet werden.

Die Benutzeroberfläche könnte, neben den im Abschnitt 7.1 genannten Aspekten, durch folgende Maßnahmen verbessert werden:

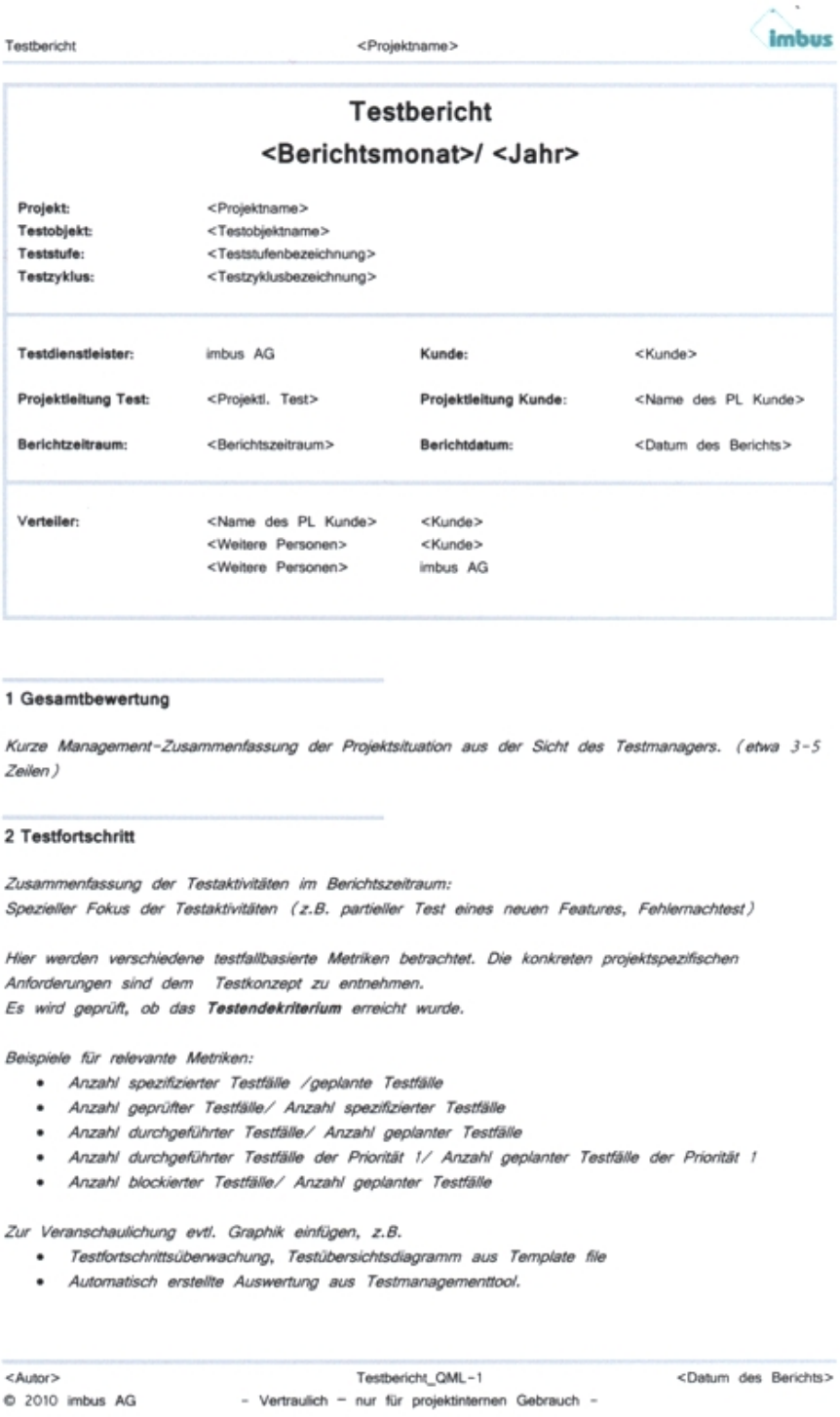
- Zeilen von Tabellen mit alternierenden, ähnlichen Hintergrundfarben geringer Sättigung anzeigen
- Sortierung der Inhalte von Tabellen benutzergesteuert durch Klick auf die Spaltenüberschrift zulassen
- Felder, die auszufüllen sind, standardmäßig mit Hinweisen füllen
- Fehlermeldungen in den nicht modalen Fenstern direkt neben den Feldern anzeigen, die Auslöser für die Fehlermeldungen waren

Die Erstellung einer Hilfe für die Anwendung bietet sich ebenfalls an. Bei der Auswahl der zu integrierenden Ideen muss darauf geachtet werden, das System weiterhin einfach zu halten und nur Funktionen umzusetzen, die wirklich benötigt werden.

In Softwareprojekten spielt das Testen eine wichtige Rolle. Für die Arbeit im Team ist es wichtig, einen Überblick über die von den Teammitgliedern erstellten Testfälle und Testdurchführungen zu erhalten. Mit Hilfe des Testfallverwaltungssystems und der darin recherchierbaren Testfälle können grundlegende Funktionen eines SUT schnell getestet werden. In der Diplomarbeit wurden theoretische und praktische Voraussetzungen geschaffen, ein System zur Verwaltung von Testfällen in der Produktentwicklung robotron*ecount einzusetzen. Nach der Weiterentwicklung des Prototyps ist es ratsam, durch die Ergebnisse eines Pilottests über die Einführung des Werkzeugs zu entscheiden und einen möglichen Schulungsaufwand abzuschätzen.

Anlagen

A Testbericht



Testbericht <Projektname> imbus

Testbericht
<Berichtsmonat>/ <Jahr>

Projekt:	<Projektname>
Testobjekt:	<Testobjektname>
Teststufe:	<Teststufenbezeichnung>
Testzyklus:	<Testzyklusbezeichnung>

Testdienstleister:	imbus AG	**Kunde:**	<Kunde>
Projektleitung Test:	<Projektl. Test>	**Projektleitung Kunde:**	<Name des PL Kunde>
Berichtzeitraum:	<Berichtszeitraum>	**Berichtdatum:**	<Datum des Berichts>

Verteiler:	<Name des PL Kunde>	<Kunde>
	<Weitere Personen>	<Kunde>
	<Weitere Personen>	imbus AG

1 Gesamtbewertung

Kurze Management-Zusammenfassung der Projektsituation aus der Sicht des Testmanagers. (etwa 3-5 Zeilen)

2 Testfortschritt

Zusammenfassung der Testaktivitäten im Berichtszeitraum:
Spezieller Fokus der Testaktivitäten (z.B. partieller Test eines neuen Features, Fehlernachtest)

Hier werden verschiedene testfallbasierte Metriken betrachtet. Die konkreten projektspezifischen Anforderungen sind dem Testkonzept zu entnehmen.
Es wird geprüft, ob das ***Testendekriterium*** *erreicht wurde.*

Beispiele für relevante Metriken:

- *Anzahl spezifizierter Testfälle /geplante Testfälle*
- *Anzahl geprüfter Testfälle/ Anzahl spezifizierter Testfälle*
- *Anzahl durchgeführter Testfälle/ Anzahl geplanter Testfälle*
- *Anzahl durchgeführter Testfälle der Priorität 1/ Anzahl geplanter Testfälle der Priorität 1*
- *Anzahl blockierter Testfälle/ Anzahl geplanter Testfälle*

Zur Veranschaulichung evtl. Graphik einfügen, z.B.

- *Testfortschrittsüberwachung, Testübersichtsdiagramm aus Template file*
- *Automatisch erstellte Auswertung aus Testmanagementtool.*

<Autor> Testbericht_QML-1 <Datum des Berichts>
© 2010 imbus AG - Vertraulich – nur für projektinternen Gebrauch -

3 Fehlerstatus

Hier werden verschiedene fehlerbasierte Metriken betrachtet. Die konkreten projektspezifischen Anforderungen sind dem Testkonzept zu entnehmen.
Ziel der Metriken soll die Ableitung eines ***Fehlertrends*** *sein.*
Es wird geprüft, ob das ***Testendekriterium*** *erreicht wurde.*

Beispiele für relevante Metriken:

- *Anzahl der gefundenen Fehler im Berichtzeitraum*
- *Anzahl der Fehler pro Testzyklus*
- *Anzahl der Fehler mit Priorität 1*
- *Anzahl der gelösten Fehler*

Zur Veranschaulichung evtl. Graphik einfügen, z.B.

- *Automatisch erstellte Auswertung aus Testmanagementtool oder Fehlermanagementtool*
- *Aus einfachem Excel-Sheet*
- *Word-Diagramm*

4 Risiken

Aktualisierte Bewertung der bereits im Testkonzept dokumentierten Risiken:

- *Hat sich die Wahrscheinlichkeit dafür, dass ein Risiko eintritt, geändert?*
- *Gibt es Indikatoren, die darauf schließen lassen, dass ein Risiko eintritt?*

Identifikation neuer Risiken:

- *Gibt es neue Risiken, die bei der Risikoidentifikation übersehen wurden?*
- *Oder, wenn es bisher keine Risikobetrachtung gibt:*
  Ist eine Situation im Projekt aufgetreten, die dazu führen kann, dass Kosten/Termine überschritten werden oder eine schlechtere Produktqualität als erwartet eintreten kann?

Risikobewältigung:

- *Gibt es bereits definierte Gegenmaßnahmen zur Minderung eines Risikos? Wurden diese eingeleitet?*
- *Oder, wenn es bisher keine Risikobetrachtung gab:*
  Vorschläge zur Risikominderung machen oder über bereits eingeleitete Maßnahmen berichten.
- *Verursachen die Maßnahmen zur Risikobewältigung zusätzliche Kosten oder Terminverschiebungen?*
- *Sind entsprechende Budgeterweiterungen beantragt/erfolgt?*
- *In welcher Höhe?*

Beispiele für mögliche Risiken/Checkliste,
siehe „Praxiswissen Softwaretest, Testmanagement", Kapitel 9.3 Risikoidentifikation.

5 Ausblick

- *Geplante Tätigkeiten im folgenden Berichtszeitraum*
- *Fokus der nächsten Testaktivitäten*
- *Abweichungen in der Planung durch*

- *Terminverschiebungen, die sich aus Verzögerungen von Vorgängeraktivitäten ergeben*
- *Terminverschiebungen, die sich aus zusätzlichen Tätigkeiten ergeben*
- *Terminverschiebungen, die sich aus Personalmangel ergeben*
- *Terminverschiebungen, die sich durch das Fehlen anderer Ressourcen ergeben*

6 Kundenzufriedenheit

An dieser Stelle bitten wir um Ihr Feedback zum bisherigen Projektverlauf:

O Alles in Ordnung.

O Der folgende Aspekt sollte künftig besser berücksichtigt werden:

O Es besteht Klärungsbedarf. Ein persönliches Gespräch sollte stattfinden.

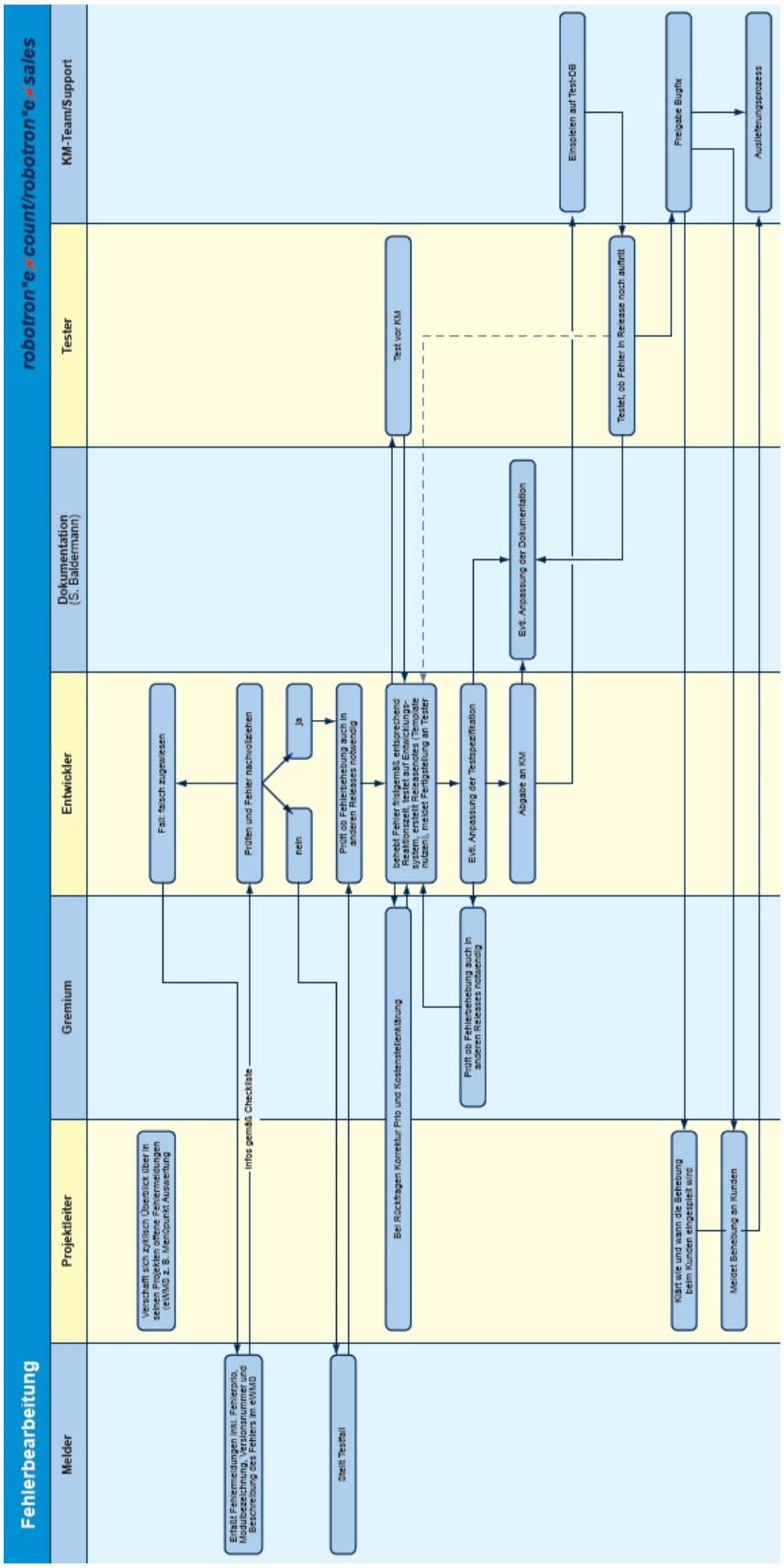
<Ort Kunde>, den .....

<Ort Dienstleister>, den <Datum des Berichts>

<Name des PL Kunde>,
Projektleitung <Kunde>

<Projektl. Test>,
Projektleitung imbus AG

B Prozesse Fehlerbearbeitung



C Bewertungsmatrix

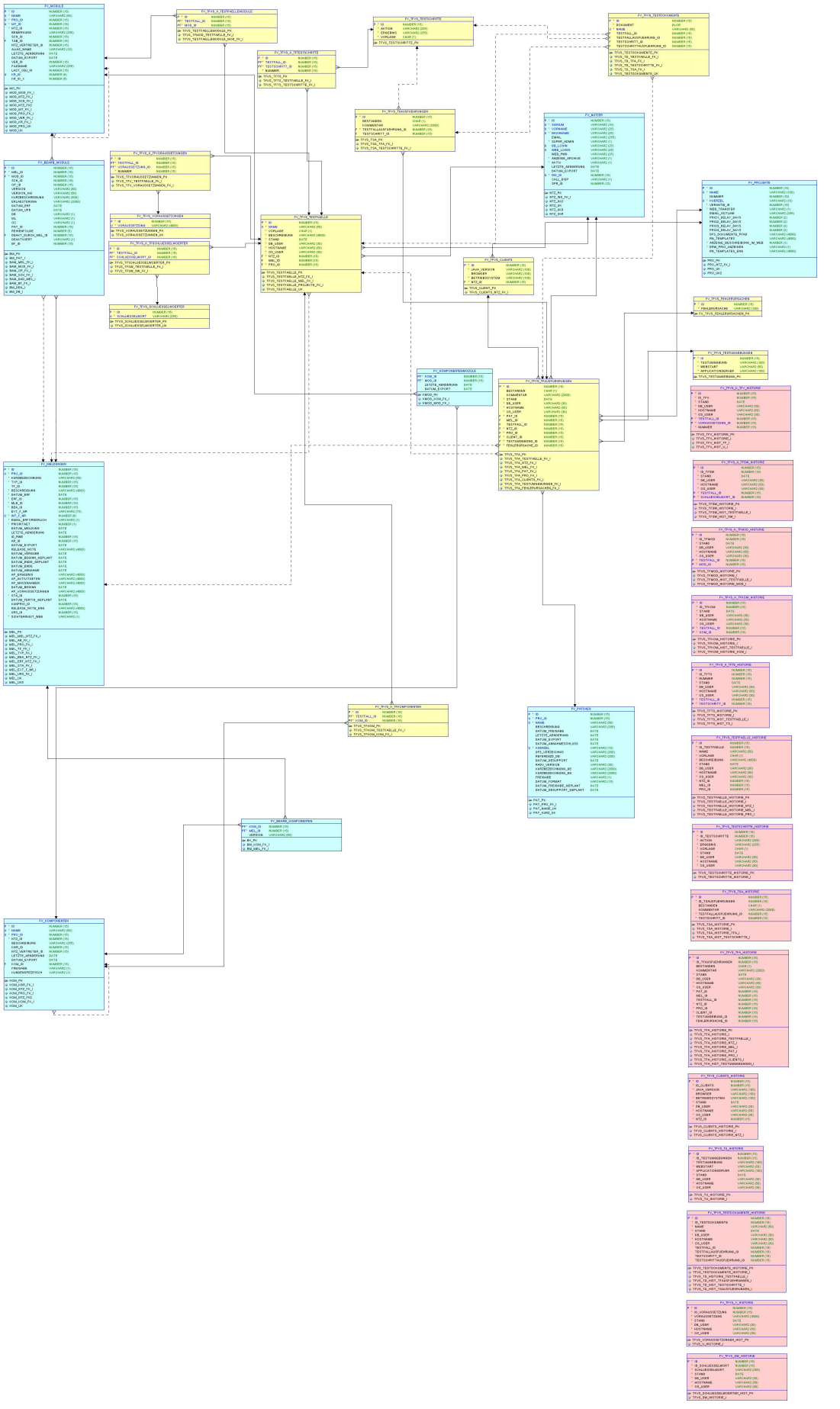
	HP Quality Ce	IBM Rationa	SilkCentral T	XQual	XStud	TestLink	Rth	QaTraq	
									Bewertung (1 (sehr gut) ... 5 (nicht ausreichend))
									Zeitmessung (in sec)
									sonstige Angaben
Bedienbarkeit									
Übersicht	2	2	3		3	2	2	2	2
Handhabung *(die Art und Weise, wie man etwas benutzt)*	2	2	2		3	2	2	2	1
Design / Oberfläche	3	1	2		2	1	3	1	
User-Verwaltung									
Rechtevergabe	3	2	1		1	2	2	1	
Benutzerangaben	2	1	1		2	2	2	2	
Testszenarien									
Struktur	2	1	2		2	2	3	1	
Testfälle									
Verwaltung	2	1	2		2	2	2	2	
Verwendete Datenfelder *(Können alle Kriterien der Testprotokolle übernommen werde*	1	1	2		3	2	1	3	
Reporting									
Gesamtübersicht	2	2	1		2	2	2	1	
Dokumentation	2	1	3		2	2	3	3	
Zeitmessung									
Benutzer anlegen	40	30	60		45	45	40	35	
Benutzerpasswort ändern	30	25	40		35	35	20	35	
Modul anlegen *(als Ordner oder Keyword / implementierte Funktionalität)*	25	55	50		110	30	35	40	
Version / Release anlegen	35	35 -	-			20	0	0	
TestCase anlegen	450	715	515		85	820	445	80	
TestCase zuordnen	20 -	-	-		-	-	-		
Test ausführen	110	90	95		100	50	100	125	
Löschen	0	0	0		0	0	0	0	
Gesamt	710	950	760		375	1000	640	315	
aufgetretene Probleme	2 (nicht alle 3 (Artikel ui		0		0	0	0	0	
Softwarequalität / Stabilität	2	2	2		2	3	2	3	
Performance *(Leistung)*	3	2	2		4	2	1	1	
sonstige Bemerkungen	läuft nicht v	anpassbar;	Versionsve		zusätzliche	Eingabeauf	Expected Result ist Pflichtfeld		
	2,15	1,83	1,89		2,11	1,98	1,86	1,5	

D Personentage

	Rth		Eigenentwicklung
Testfälle Komponenten / Modulen zuordnen	2		6,75
Testprotokolle in Ordner ablegen	0,5		1,75
Oracle DB nutzen	4		0
Dokumente zu Testfällen hinzufügen	0		1,5
Testausführung in Steps	0		2,5
Testfälle verfassen, editieren, entfernen	0		4,25
Versionskontrolle für Testfälle	17		3
Ressourcenkonfigurationen eintragen	0		0
Statistiken	2		3
Besitzer	0		1,5
Templates	0,5		1
Integration	0		0
Einarbeitung	3		5
GESAMT	**29**		**30,25**

E Datenmodell

Die Tabellen des Datenmodells sind farbig gekennzeichnet. Tabellen mit der Hintergrundfarbe Blau sind bereits im Fehlerverwaltungssystem robotron*eWMS vorhanden und sollen von dem Testfallverwaltungssystem benutzt werden. Die Tabellen mit der Hintergrundfarbe Gelb und die Tabellen mit der Hintergrundfarbe Rot sollen für das Testfallverwaltungssystem neu angelegt werden. Rot gekennzeichnet sind Tabellen, die für die Historisierung verwendet werden sollen. Sie besitzen keine Fremdschlüssel und keine Unique Keys.



Literaturverzeichnis

[Bar92] Barker, Richard. Case method: Entity-Relationship-Modellierung. Addison-Wesley, 1992.

[Fru07] Frühauf, Karol; Ludewig, Jochen; Sandmayr, Helmut. Software-Prüfung. vdf, 2007.

[Lig09] Liggesmeyer, Peter. Software-Qualität. Spektrum Akademischer Verlag, 2009.

[Spi08] Spillner, Andreas; Roßner, Thomas; Linz, Tilo. Praxiswissen Softwaretest. dpunkt.verlag, 2008.

[Spi10] Spillner, Andreas; Linz, Tilo. Basiswissen Softwaretest. dpunkt.verlag, 2010.

[Tid06] Tidwell, Jenifer. Designing Interfaces. O'Reilly, 2006.

[Vos08] Vossen, Gottfried. Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme. Oldenbourg Verlag München Wien, 2008.

[Wal90] Wallmüller, Ernest. Software-Qualitätssicherung in der Praxis. Hanser, 1990.

[IEE98] IEEE Std 829-1998. IEEE Standard for Software Test Documentation.

[WID10] Relationale Datenbank – Wikipedia. http://de.wikipedia.org/wiki/Relationenmodell, 10 2010.

[WIF10] Oracle Forms – Wikipedia. http://de.wikipedia.org/wiki/Oracle_Forms, 10 2010.

[WIK10] KISS-Prinzip – Wikipedia. http://de.wikipedia.org/wiki/KISS-Prinzip, 10 2010.

[WIO10] Topologische Sortierung – Wikipedia. http://de.wikipedia.org/wiki/Topologische_Sortierung, 10 2010.

[WIP10] PL/SQL – Wikipedia. http://de.wikipedia.org/wiki/PL/SQL, 10 2010.

[WIR10] Relation (Datenbank) – Wikipedia. http://de.wikipedia.org/wiki/Relation_(Datenbank), 10 2010.

[WIT10] Testumgebung – Wikipedia. http://de.wikipedia.org/wiki/Testumgebung, 10 2010.

[CRE10] Glossar zur Konfiguration - CREALIS. http://www.crealis.de/de/crealis/glossar.html, 10 2010.

[IAG10] imbus AG - Beratung. http://www.imbus.de/downloads/beratung/, 10 2010.

[MKN10] Glossar. http://www.management-knowhow.de/index.php?id=10, 10 2010.

[MOT10] Modellbasiertes Testen – Winfwiki. http://winfwiki.wi-fom.de/index.php/Modellbasiertes_Testen, 10 2010.

[NNB10] Gewusst wie: Zuordnen von n:n-Beziehungen. http://msdn.microsoft.com/de-de/library/d8b1fb61%28VS.80%29.aspx, 10 2010.

[PMG10] Siegfried Seibert: Wissensspeicher.Projektmanagement-Glossar. http://www.siegfried-seibert.de/Wissensspeicher/PMGlossar, 10 2010.

[QHB10] Organisationshandbuch: Qualitative Bewertungsmethoden. http://www.orghandbuch.de/nn_414926/OrganisationsHandbuch/DE/6__MethodenTechniken/65__Bewertungsverfahren/652__Qualitative/qualitative-node.html?__nnn=true, 10 2010.

[RDS10] Software |mit Energie. http://download.robotron.de/pdf/flyer_robotron_energie_0110.pdf, 01 2010.

[SKM08] Software-Konfigurationsmanagement. http://swt.cs.tu-berlin.de/lehre/sepr/ws0708/referate/SCM_Ausarbeitung.pdf, 2008.

[WMO03] Das W-Modell. http://www.gi-hb-ol.de/uta/gi-rg/WModellSpillner.pdf, 2003.

[HPQ10] HP Quality Center software - HP - BTO Software. https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-127-24_1131_4000_100__, 10 2010.

[IBM10] Rational Quality Manager and Rational Test Lab Manager 2.0 Downloads - Jazz Community Site. https://jazz.net/downloads/rational-quality-manager/releases/2.0?p=allDownloads&S_CMP=web_dw_rt_swd_bod, 10 2010.

[QAT10] Test Management. http://www.testmanagement.com/download.html, 10 2010.

[RTH10] RTH - Requirements and Testing Hub | Download RTH - Requirements and Testing Hub software for free at SourceForge.net.
http://sourceforge.net/projects/rth/, 10 2010.

[SCT10] SilkCentral Test Manager Product Trial - Micro Focus.
http://www.microfocus.com/products/silk/SilkCentral-Test-Manager-product-trial.aspx, 10 2010.

[TSL10] Browse TestLink Files on SourceForge.net.
http://sourceforge.net/projects/testlink/files/, 10 2010.

[XQA10] Download XStudio. http://www.xqual.com/support/download.html, 10 2010.

Erklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Dresden, 19. Oktober 2010